People's Democratic Republic of Algeria Ministry of Higher Education and Scientific Research

University M'hamed Bougara - Boumerdes

Institute of Electrical and Electronic Engineering

Project Report Presented in Partial Fulfillment of the Requirements of the Degree of:

'Master'

In Computer Engneering

Machine Learning-Based Approaches for Intrusion Detection

Presented by :

Soulef KEZZOULA

Supervisor :

Mr. A. MOHAMMED-SAHNOUN

Promotion: 2022/2023

Dedication

“

To my incredible father, I am forever grateful for your wisdom, patience, support, and guidance that have shaped my academic and personal growth. Thank you, Dad, for being my mentor, and my cheerleader throughout this journey.

To my amazing mother, I am eternally grateful for your consistent presence in my life. Your love and strength, have been a constant source of inspiration . Thank you, Mom, for being my role model, and my confidante.

To my dear grandmother, though you are no longer with us, in our hearts you hold a place, no one else will ever fill. In life, we loved you dearly, In death, we love you still.

To all my beloved ones

Thank you.

”

Acknowledgment

First and foremost, I would like to thank ALLAH the Almighty who provided me with his blessing and the opportunity to successfully conclude this project.

In the accomplishment of my final year project, I would like to express my deepest gratitude to my supervisor **Mr.A. MOHAMMED-SAHNOUN**, for his invaluable guidance, unwavering support and patience during this long journey, I am very thankful for all of his information, remarks, and encouraging words. I had the honor to share my project with him.

I would like to extend my heartfelt appreciation to my colleague, **Mounir OUADI**, for his valuable contribution and support throughout this project. His willingness to share his expertise, provide feedback, and engage in fruitful discussions has greatly enriched the project.

Last but not least, I would like to thank all people who have helped me and assisted me throughout this journey. Their guidance have been invaluable, and I am deeply grateful for their help and contributions.

Abstract

The rapid increase in network security threats necessitates the development of effective intrusion detection model capable of identifying and mitigating malicious activities. This approach presents the development and evaluation of an intrusion detection model utilizing various machine learning algorithms that provide a potential pathway for attacks detection. The model is designed to detect and mitigate malicious activities within a network environment. To train and test the machine learning models, a comprehensive dataset sourced from Kaggle is employed, encompassing a wide range of attack scenarios and normal network behaviors. The models are evaluated using classification evaluation metrics such as accuracy, precision, and recall, demonstrating their capability to accurately classify instances and identify potential threats. The primary objective of this approach is to enhance the detection of attacks irrespective of their types, while minimizing the occurrence of false positives. The results obtained highlight the effectiveness of the intrusion detection model in providing a proactive network security solution, enabling timely detection and response to malicious activities. The findings of this project contribute to the field of intrusion detection, paving the way for improved security measures in network environments.

Keywords : Intrusion detection, Machine learning, Classification, Security.

Contents

List of Figures

List of Tables

Liste of Abreviations

ID *Intrusion Detection*

IDS *Intrusion Detection System*

HIDS *Host Based IDS*

NIDS *Network Based IDS*

APIDS *Applicaion Protocol based IDS*bas

NNIDS *Network Node IDS*bas

ArachNIDS *Advanced Reference Archive of Current Heuristics for Network Intrusion*bas

MIDAS *Multics Intrusion Detection and Alerting System*bas

DOS *Denail Of Service*

DDoS *Distributed Denial Of Service*

ML *Machine Learning*

SVMs *Support Vector Machines*

KNN *K-Nearest Neighbor*

KDD *Knoweldge Discovery and Data mining*

NSL	*Network Security Laboratory*
API	*Application Programming Interface*
Colab	*Colaboratory*
ROC	*Receiver Operating Characteritic Curve*
TN	*True Negative*
FP	*False Positive*
FN	*False Negative*
FPR	*False Positive Rate*
TPR	*True Positive Rate*
AUC	*Area Under Curve*

General Introduction

Context

In today's interconnected world, the security of computer networks is of utmost importance. With the increasing sophistication of cyber-attacks and the potential for significant damage, organizations are actively seeking advanced solutions to protect their systems and sensitive data. One such solution is an Intrusion Detection System (IDS), which plays a crucial role in identifying and mitigating potential security breaches. Traditionally, IDSs relied on rule-based approaches and signature-based detection methods, which were limited in their ability to handle evolving and complex attack patterns. However, with the advancements in machine learning techniques, IDSs can now leverage the power of artificial intelligence to detect and respond to a wide range of network intrusions. The primary objective of this project is to design and implement an Intrusion Detection System using machine learning algorithms.

Problematic and Objectives

The ever-increasing sophistication of cyber-attacks poses a significant challenge for organizations in protecting their computer networks and sensitive data. Traditional rule-based and signature-based Intrusion Detection Systems (IDSs) often struggle to keep up with the evolving attack techniques, leading to potential security breaches and data loss. Therefore, there is a critical need for advanced intrusion detection techniques that can effectively detect and respond to emerging threats in real-time. The problem addressed in this project is to design and implement an efficient and accurate Intrusion Detection System using machine learning algorithms. The goal is to develop a system that can

autonomously analyze network traffic, identify anomalous behavior, and classify it as either normal or malicious. By leveraging the power of machine learning, we aim to improve the detection capabilities of IDSs, reducing false positives and false negatives, and enabling timely response to potential security incidents.

Report organization

This project is organized into three main chapters, each focusing on a specific aspect of the research topic "Machine Learning-Based approaches for Intrusion Detection".

In the first chapter, we will present a general study about Intrusion Detection System (IDS), importance in network security, and the challenges faced in detecting and preventing intrusions. We will Discuss different types of IDS, including Host based IDS, Network based IDS, and Application based IDS, highlighting their strengths and limitations. We Provides an overview of common network attacks, such as Denial of Service, Botnet and Virus. In the final section, we will address the evolution of Intrusion Detection Systems.

In the second chapter, we provide a comprehensive introduction to machine learning, its principles, and its role in building intelligent systems. We discuss the two main types of machine learning: supervised and unsupervised learning. We review also various supervised learning algorithms, such as logistic regression, decision trees, random forest, and k-nearest neighbors (KNN), highlighting their characteristics and applications. We will even cover unsupervised learning algorithms, including clustering algorithm and association rule mining.

The last chapter includes the data exploration and the data preprocessing part where we will describe the NSL-KDD dataset in the study, its features, and statistical analysis of the data. Provides insights into the distribution of features, class imbalance, and correlations. We Discuss the steps taken to preprocess the dataset, including handling missing values, scaling numerical features, encoding categorical variables, and handling class imbalance. We address to the results and discussions obtained from the models described in the second chapter and in the last part, we will have our summary of findings obtained from the research.

Chapter 1

Introduction to Intrusion Detection System

1.1 Introduction

In the current chapter, we will explain several terms and definitions relations to our theme that we found necessary to know for a good understanding of the subject.

The figure 1.1 shows an example of network that can be attacked:

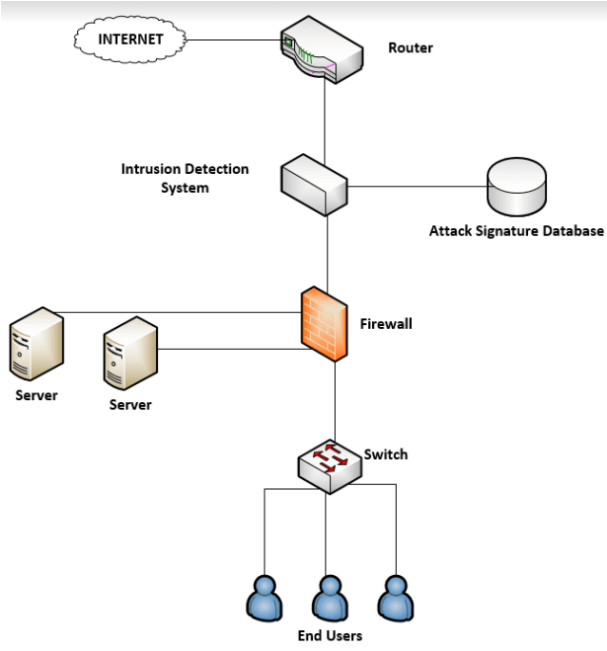

Figure 1.1: Attack Network [1]

1.2 Definitions

1.2.1 Intrusion

Intrusion refers to the act of unauthorized access, entry, or penetration into a computer system, network, or device. An intrusion can be intentional or unintentional and may involve theft, damage, or alteration of data or resources. Intrusions can be carried out by individuals, groups, or automated systems, such as malware or viruses. The detection and prevention of intrusions are important for maintaining the security and integrity of computer systems and networks[2].

1.2.2 Intrusion Detection

Intrusion detection is the process of identifying unauthorized access or activity on a computer system or network. It involves monitoring the system or network for suspicious behavior, such as attempts to access restricted resources or changes to system configurations. It is a complete security approach that provides a wide range of intrusion detection capabilities to help administrators secure and monitor their network environments against threats [3].

1.2.3 Intrusion Detection System

Intrusion Detection System (IDS) is a system that monitors and analyzes data to detect any intrusion in the system or network. Intrusion detection systems are hardware and software systems that monitor events occurred on computers and computer networks in order to analyze security problems.

An intrusion detection system is referred as burglar alarm. For example the lock system in the house protects the house from theft. But if somebody breaks the lock system and tries to enter into the house, it is the burglar alarm that detects that the lock has been broken and alerts the owner by raising an alarm. The number and severity of these attacks has been increasing continuously. Consequently Intrusion detection systems have become an integral part of the security infrastructure of organizations [4].

1.3 Intrusion detection types

IDS can be classified into three major categories:

- Host based IDS (HIDS)
- Network based IDS (NIDS)
- Application based IDS (APIDS)

The figure 1.2 shows Intrusion Detection types diagram:

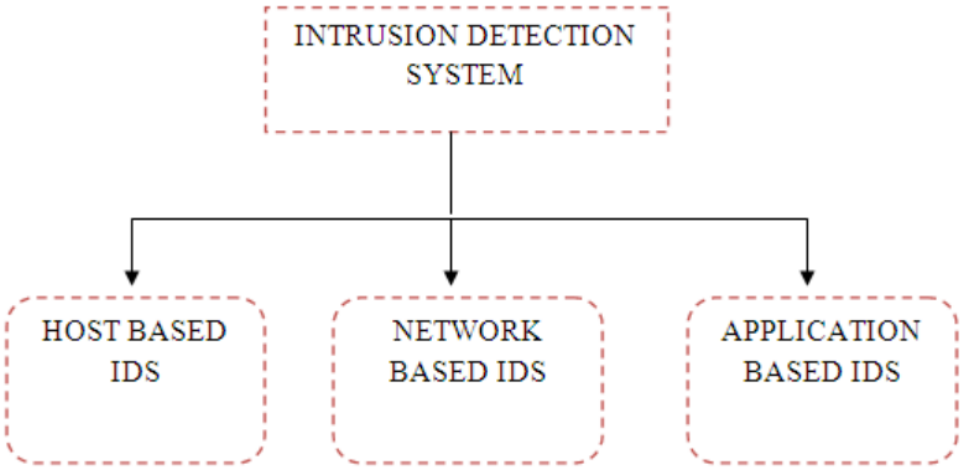

Figure 1.2: Intrusion Detection System Types [5]

1.3.1 Host based IDS

Host based IDS views the sign of intrusion in the local system. For analysis, they use host system's logging and other information. Host based handler is referred as sensor. Other sources, from which a host-based sensor can obtain data, include system logs and other logs generated by operating system processes and contents of objects not reflected in standard operating system audit and logging mechanisms. Host based system trust strongly on audit trail.

The information allows the intrusion detection system to spot subtle patterns of misuse that would not be visible at a higher level of abstraction. A host-based IDS provides much more relevant information than Network-based IDS.

HIDS are used efficiently for analyzing the network attacks, for example, it can sometimes tell exactly what the attacker did, which commands he used, what files he opened, rather than just a vague accusation and there is an attempt to execute a dangerous command. It is less risky to configure [2].

Advantages of Host based IDS

- It verifies success or failure of an attack
- Monitors System Activities

- Detects attacks that a network based IDS fail to detect
- Near real time detection and response
- Does not require additional hardware
- Lower entry cost [6]

The figure 1.3 shows a Host-based Intrusion Detection System:

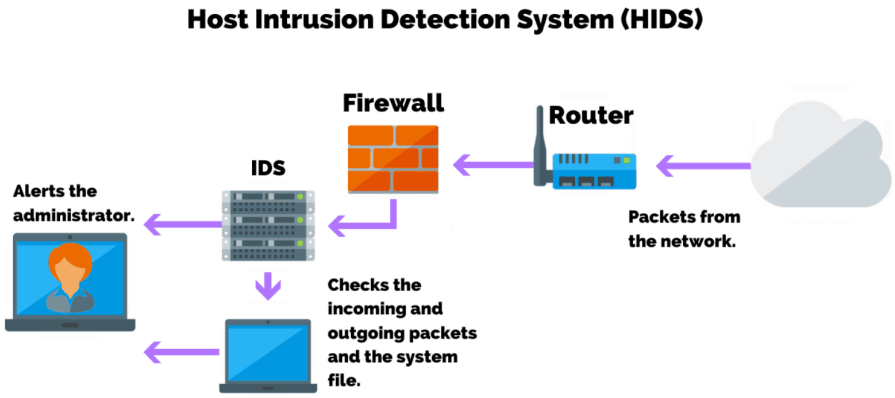

Figure 1.3: Host-based Intrusion Detection System [7]

1.3.2 Network based IDS

Network based IDS systems collect information from the network itself rather than from each separate host. The NIDS audits the network attacks while packets moving across the network. The network sensors come equipped with attack signatures that are rules on what will constitute an attack and most network-based systems allow advanced users to define their own signatures. Attack on the sensor is based on signature and they are from the previous attacks and the operation of the monitors will be transparent to the users and this is also significant.

The transparency of the monitors decreases the likelihood that an adversary will be able to locate it and nullify its capabilities without the efforts. Network Node IDS (NNIDS) agents are deployed on every host within the network being protected [8].

The figure 1.4 shows a Newtork-based Intrusion Detection System:

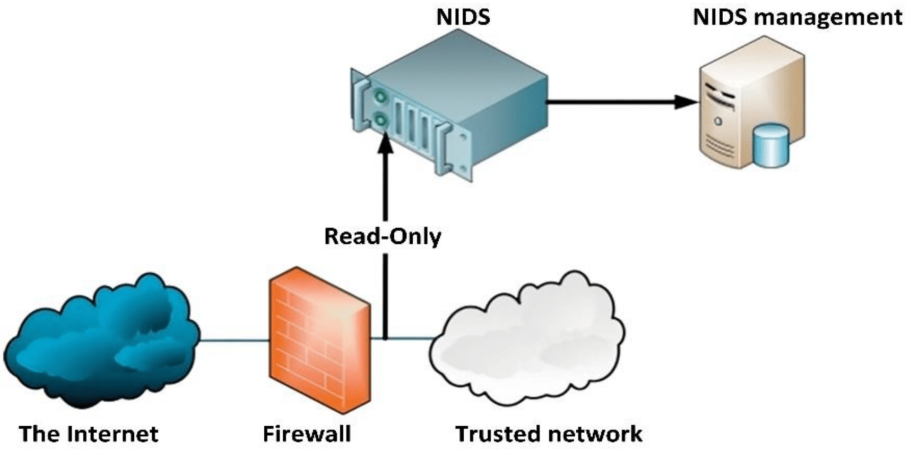

Figure 1.4: Network-based Intrusion Detection System [9]

1.3.3 Application Protocol based IDS

Application Protocol based IDS will check the effective behavior and event of the protocol. The system or agent is placed between a process and group of servers that monitors and analyzes the application protocol between devices [10].

1.4 Intrusion Detection Methods

We have two major categories of intrusion detection systems, which include signature based intrusion detection and anomaly based intrusion detection.

1.4.1 Signature based Intrusion Detection method

Signature based intrusion detection is termed as misuse detection. In this detection approach, user's activities are compared with the attackers' known behaviors, to penetrate a system or network. In misuse detection, gathered information is analyzed and compared with large databases for attack signatures. The positive side of misuse IDS is the ability to detect known attacks with great precision. In general, they have a high rate of detection and low rate of false alarms compared to anomaly-based systems.

One approach to this detection is the use of arachNIDS (Advanced Reference Archive of Current Heuristics for Network Intrusion) which has a small signature for each attack[11].

The figure 1.5 shows the Signature and Anomaly-Based Intrusion Detection system diagram:

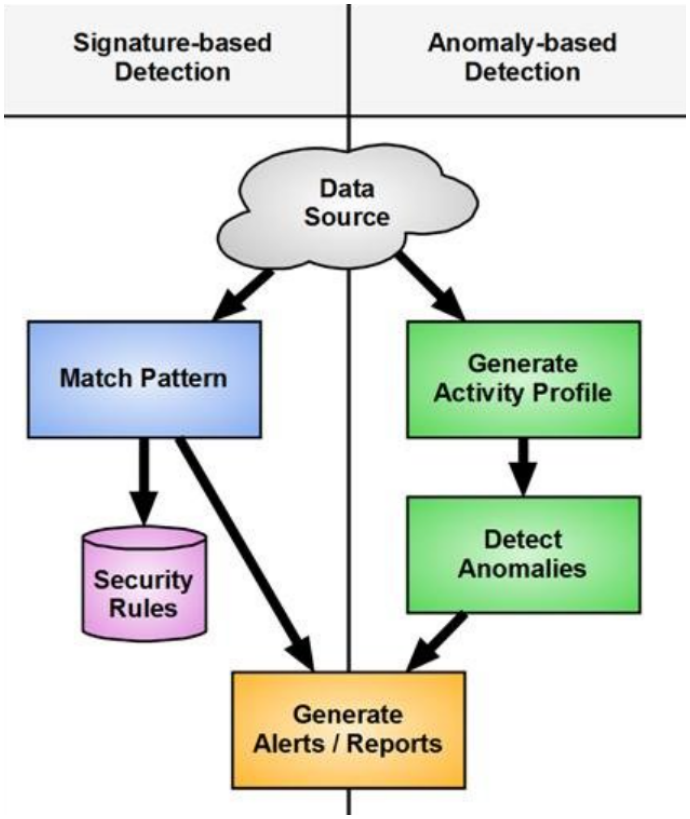

Figure 1.5: Signature, Anomaly-Based IDS [12]

1.4.2 Anomaly based Intrusion Detection method

The anomaly-based technique stores the normal behavior of a user in a database and compares it with the current behavior of the user. If there is a substantial difference, then there is something wrong or abnormal. The major advantage of anomaly detection is that it does not require information of known attacks, and thus they can detect new forms of attacks. It has a high rate of false alarm compared to misuse-based IDS.

One approach to this detection is the use of MIDAS (Multics Intrusion Detection and Alerting System), it's an expert system which consists a set of knowledge base and set of rules. Each time a user action occurs, it will be added to the knowledge base [13].

1.5 Common Types of Network Attacks

Without security measures and controls in place, your data might be subjected to an attack. Some attacks are passive, meaning information is monitored; others are active, meaning the information is altered with intent to corrupt or destroy the data or the network itself. In the rest of this section, we present some network attacks [14].

1.5.1 Denial Of Service DOS

Denial of Service (DoS) is a type of cyber-attack that aims to disrupt the normal functioning of a computer system, network, or website. The goal of a DoS attack is to overwhelm the targeted system with a flood of traffic, requests, or data, making it unable to respond to legitimate requests and causing it to crash or become unavailable [15].

DoS attacks can be launched in different ways, including flooding a network with traffic from multiple sources (Distributed Denial of Service or DDoS), exploiting vulnerabilities in software or hardware, or sending malformed data packets to crash specific services [15].

The impacts of DoS attacks can range from minor disruptions to critical service outages, leading to financial losses, reputational damage, and legal consequences. To prevent DoS attacks, organizations need to implement robust security measures, such as firewalls, intrusion detection systems, load balancers, and content delivery networks, among others [16].

1.5.2 Botnet

Botnets are often created by infecting computers or devices with malware, which allows the attacker to take control of the infected machines remotely [17].

Once a botnet is established, the attacker can use it to perform a variety of malicious activities, such as launching Distributed Denial of Service (DDoS) attacks, sending spam or phishing emails, stealing sensitive information, or distributing more malware. Because botnets are composed of numerous machines, they can generate large volumes of traffic or spam, making them difficult to detect and mitigate.

Botnets can be created using a variety of methods, such as exploiting vulnerabilities in software or hardware, tricking users into downloading and installing malware, or using brute force attacks to guess passwords. To prevent botnets, it is important to keep software and systems up-to-date with security patches, use strong and unique passwords, and use antivirus software and firewalls to detect and block potential threats [18].

1.5.3 Virus

A computer virus is a type of malicious software that infects a computer system or device by replicating itself and modifying other software without the user's consent or knowledge. The primary goal of a computer virus is to cause harm to the system, steal sensitive data, or gain unauthorized access to resources [19].

1.6 Intrusion Detection System Hardware:

An intrusion detection system (IDS) is a security solution that can take the form of a hardware device or a software application, operating in the application layer of the network, it detects real-time traffic and searches for attack signatures or traffic patterns, then sends out alarms and this is done through [20]:

- System file comparisons against malware signatures.
- Scanning processes that detect signs of harmful patterns.
- Monitoring user behavior to detect malicious intent.
- Monitoring system settings and configurations.

The figure 1.6 shows an hardware architecture of Intrusion Detection System:

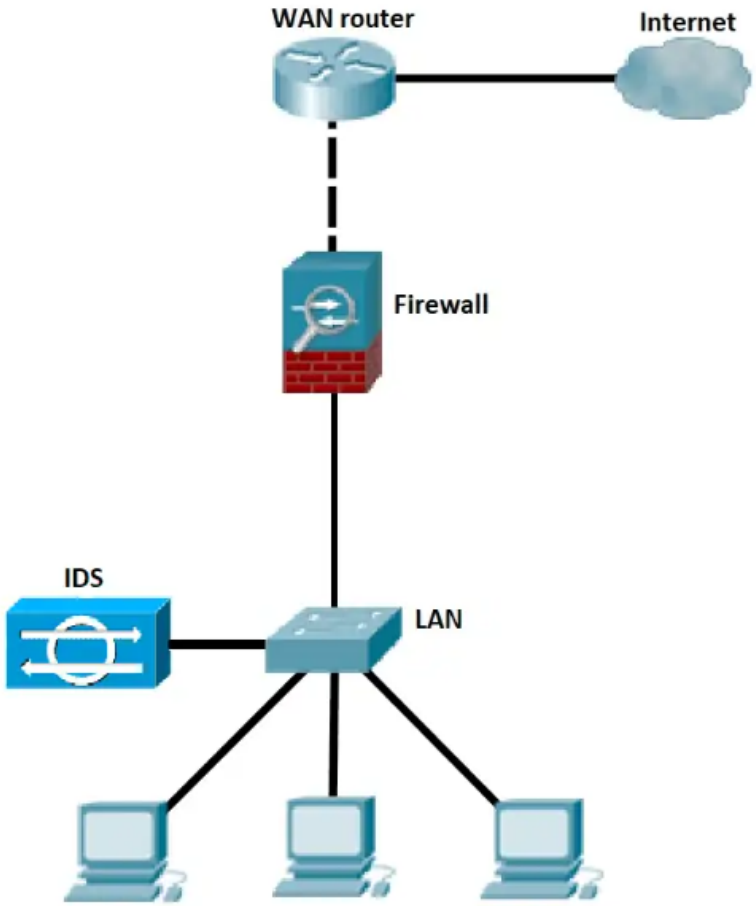

Figure 1.6: Hardware architecture of Intrusion Detection System [21]

1.7 Intrusion Detection System Evolution

In the last few years, the ID field has grown considerably and therefore a large number of IDS have been developed to address specific needs. The initial ID systems were once anomaly detection tools but today, misuse detection tools dominate the market. With an increasingly growing number of computer systems connected to networks, ID has become a necessity. To help solve the knowledge-based problems, workshops have been held every year for the past four years to share information related to ID.

The as we topics are quite varied every year and they cover a wide range of subjects such as Lesson Learned, IDS and Law, Modeling Attacks, Anomaly Detection, etc. These

workshops main objective are to find new solutions to new and challenging problems. The problems, the as we community are now facing are high-speed networks and switching.

The goal is to frustrate attackers by using an IDS architecture invisible to attackers' normal means of mapping a network. The most common way of accomplishing this "invisibility" is by restricting the communication allowed between different security components on a private network [22].

1.8 Conclusion

In conclusion, the first chapter of this project provided a comprehensive overview of Intrusion Detection Systems (IDS) and various types of attacks commonly encountered in computer networks. We began by introducing the concept of intrusion detection and its significance in ensuring the security and integrity of network systems. We discussed the two primary categories of IDS: network-based intrusion detection systems (NIDS) and host-based intrusion detection systems (HIDS). Furthermore, we delved into the different types of attacks that pose threats to network security. We examined some of the most prevalent attack types, including Denial of Service (DoS) attacks, Distributed Denial of Service (DDoS) attacks, Botnet and Virus. By familiarizing ourselves with the various types of IDS and common attack vectors, we have gained a solid foundation for building effective intrusion detection systems. This knowledge will serve as a basis for the subsequent chapters, where we will delve into the application of machine learning techniques for intrusion detection. In the following chapter, we will shift our focus towards machine learning and its techniques.

Chapter 2

Introduction to machine learning

2.1 Introduction

A intrusion detection system based on anomaly detection controls system activities in order to classify them as normal or abnormal. It proceeds to build profiles of normal behavior for user activities and to observe significant deviations from the current user activity compared to the established normal form. To be able to formalize the normal behavior of a system, several methods have been used. This chapter will be devoted to a general presentation of these different approaches.

2.2 Machine Learning

Machine learning is a multi-disciplinary field having a wide-range of research domains reinforcing its existence. The simulation of ML models is significantly related to Computational Statistics whose main aim is to focus on making predictions via computers. It is also co-related to Mathematical Optimization which relates models, applications and frameworks to the field of statistics. Real world problems have high complexity which make them excellent candidates for application of ML. Machine learning can be applied to various areas of computing to design and program explicit algorithms with high performance output, for example, email spam filtering, fraud detection on social network, online stock trading, face and shape detection, medical diagnosis, traffic prediction, character recognition and product recommendation amongst others. The field of machine learning involves building a model from data using an algorithm. This model will generalize as best as possible by representing or approximating the data. Depending on the input data given to it, it can predict unknown data as well as better understand existing data. [23]

2.3 Machine learning paradigms

Depending on how an algorithm is being trained and on the basis of availability of the output while training, machine learning paradigms can be classified into ten categories but the most ones used are supervised learning and unsupervised learning which will be explained in the following subsections.

2.3.1 Supervised learning

In supervised learning, The algorithm learns to map the input data to the correct output by iteratively adjusting its internal parameters until it can accurately predict the correct output for new input data. Supervised learning is used in a variety of applications such as image recognition, natural language processing, and speech recognition. We can deduce two many types of supervised learning: Classification and regression, and several algorithms used such as decision trees, support vector machines (SVMs), and neural networks [24].

Classification

The Classification algorithm is a supervised learning technique that is used to identify the category of new observations on the basis of training data. Mathematically, a program learns from the given dataset or observations and then classifies new observation into a number of classes or groups. Such as, Yes or No, 0 or 1, Spam or Not Spam, cat or dog, etc. Classes can be called as targets/labels or categories. In classification algorithm, a discrete output function(y) is mapped to input variable(x). y= f(x), where y= categorical output. In the following, we summarize the common classification problems [25].

- Binary classification

It refers to the classification tasks having two class labels such as “true and false” or “yes and no”. In such binary classification tasks, one class could be the normal state, while the abnormal state could be another class. For example ”attack detected”, ”attack not detected” are considered as binary classification [26].

- Multi-class classification

Traditionally, this refers to those classification tasks having more than two class labels. The multi-class classification does not have the principle of normal and abnormal outcomes, unlike binary classification tasks. Multiclass classification aims to classify instances into three or more distinct classes, unlike binary classification, where the goal is to separate instances into two classes. It requires the use of algorithms and techniques that can

handle multiple classes simultaneously, enabling the prediction of the most appropriate class label for each instance based on its features. For example, in image recognition problems, the task of multiclass classification could involve identifying various objects such as cats, dogs, birds, and cars. The algorithm would analyze the features of each image and assign it to the most appropriate class label. The goal of multiclass classification is to achieve accurate and reliable classification across all classes, it facilitates in depth analysis and decision-making in complex classification problems [27].

- Multi-label classification

In machine learning, multi-label classification is an important consideration where an example is associated with several classes or labels. Thus, it is a generalization of multi-class classification, where the classes involved in the problem are hierarchically structured, and each example may simultaneously belong to more than one class in each hierarchical level, e.g., multi-level text classification. For instance, google news can be presented under the categories of a "city name", "technology", or "latest news", etc. Multi-label classification includes advanced machine learning algorithms that support predicting various mutually non exclusive classes or labels, unlike traditional classification tasks where class labels are mutually exclusive [28].

Regression

Regression analysis includes several methods of machine learning that allow to predict a continuous (y) result variable based on the value of one or more (x) predictor variables. Regression models are now widely used in a variety of fields, including financial forecasting or prediction. In the following, we will summarize the two main types of regression [26].

- Linear Regression

It is a statistical method that is used for predictive analysis. It makes predictions for continuous/real or numeric variables such as sales, salary, age, product price...and it aims to find the best-fit linear equation that describes the linear relationship between the variables. In linear regression, The dependent variable or the target or also known as the response variable, is assumed to be a linear combination of the independent variables,

which are often referred to as features or predictors. The goal is to estimate the coefficients of the linear equation that minimize the difference between the predicted values and the actual values of the dependent variable [29].

Mathematically, we can represent a linear regression as:

$$y = \beta 0 + \beta 1.x1 \tag{2.1}$$

where:

y: the dependent variable

x: the independent variable

$\beta 0$: intercept of the line, representing the value of y when x=0

$\beta 1$: s the slope, representing the change in y per unit change in x

- Logistic Regression

Logistic regression is a statistical modeling technique used to predict the probability of a binary or categorical outcome. It is commonly used for classification problems where the dependent variable or outcome variable is categorical in nature. Mathematically, We use the logistic function or the sigmoid function to calculate probability in logistic regression. The logistic function is a simple S-shaped curve used to convert data into a value between 0 and 1 [30].

The equation of logistic regression can be defined as:

$$sigmoid(z) = 1/(1 + exp(-z)) \tag{2.2}$$

The figure 2.1 shows an example of Linear Regression versus Logistic Regression:

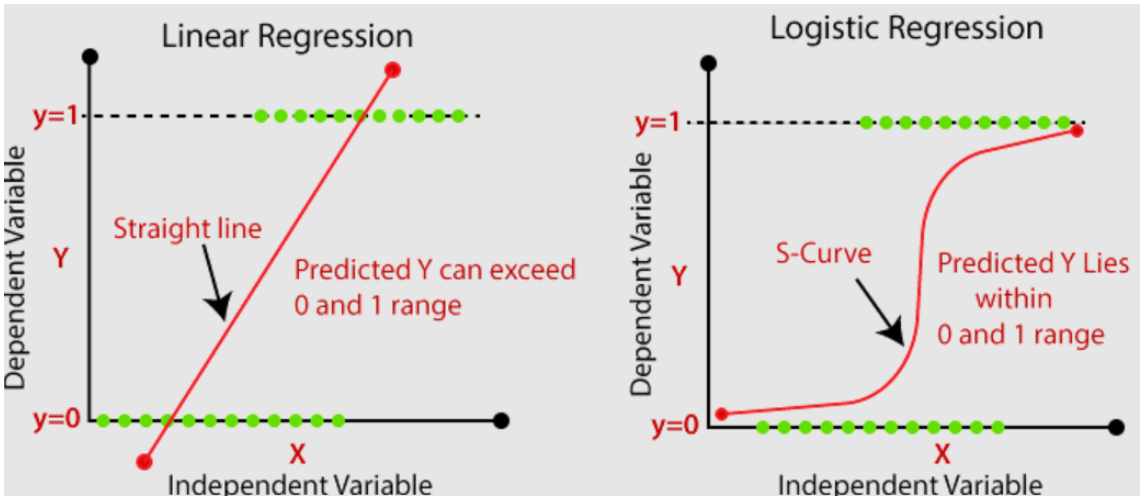

Figure 2.1: Linear Regression VS Logistic Regression [31]

Decision Tree

A decision tree is a type of supervised learning algorithm used for classification and regression. It works by recursively splitting the input data into subsets based on the values of input features, until a decision can be made on the final class or output value. In a Decision tree, there are two nodes, which are the Decision Node and Leaf Node. Decision nodes are used to make any decision and have multiple branches, whereas Leaf nodes are the output of those decisions and do not contain any further branches. A decision tree represents the minimum number of yes/no questions to be asked to make a correct decision. As a method, it allows you to approach the problem in a systematic way to arrive to a logical conclusion [32].

The figure 2.2 shows an example of Decision tree graph:

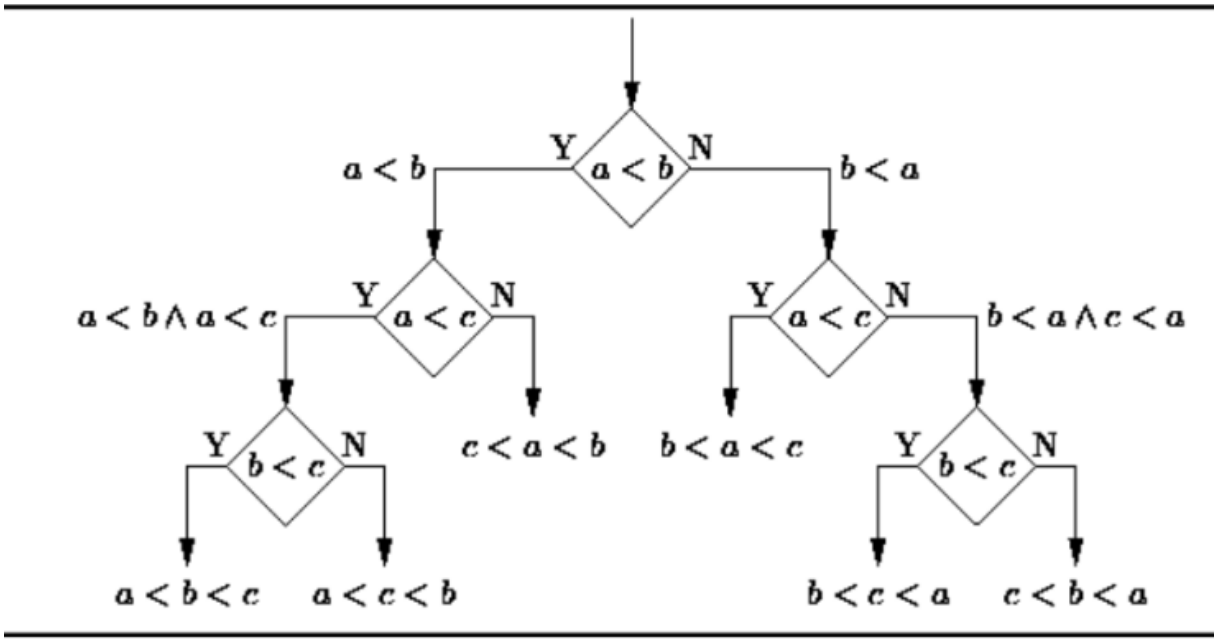

Figure 2.2: Example of Decision tree graph [33]

Support Vector Machine

Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning. The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane. SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called as support vectors, and hence algorithm is termed as Support Vector Machine. Consider the below diagram in which there are two different categories that are classified using a decision boundary or hyperplane [34].

The figure 2.3 shows an example of Support Vector Machine:

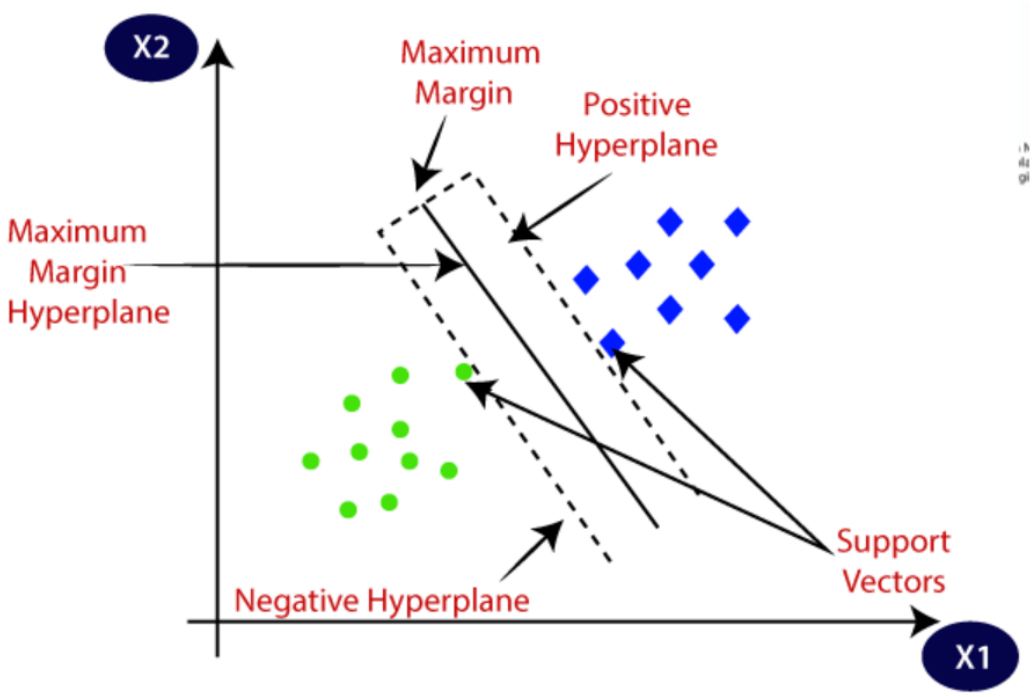

Figure 2.3: Support Vector Machine example [35]

Random forest

A Random Forest Algorithm is a supervised machine learning algorithm that is extremely popular and is used for Classification and Regression problems in Machine Learning. We know that a forest comprises numerous trees, and the more trees more it will be robust. Similarly, the greater the number of trees in a Random Forest Algorithm, the higher its accuracy and problem-solving ability. Random Forest is a classifier that contains several decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset. It is based on the concept of ensemble learning which is a process of combining multiple classifiers to solve a complex problem and improve the performance of the model [36].

The figure 2.4 shows an example of Random Forest:

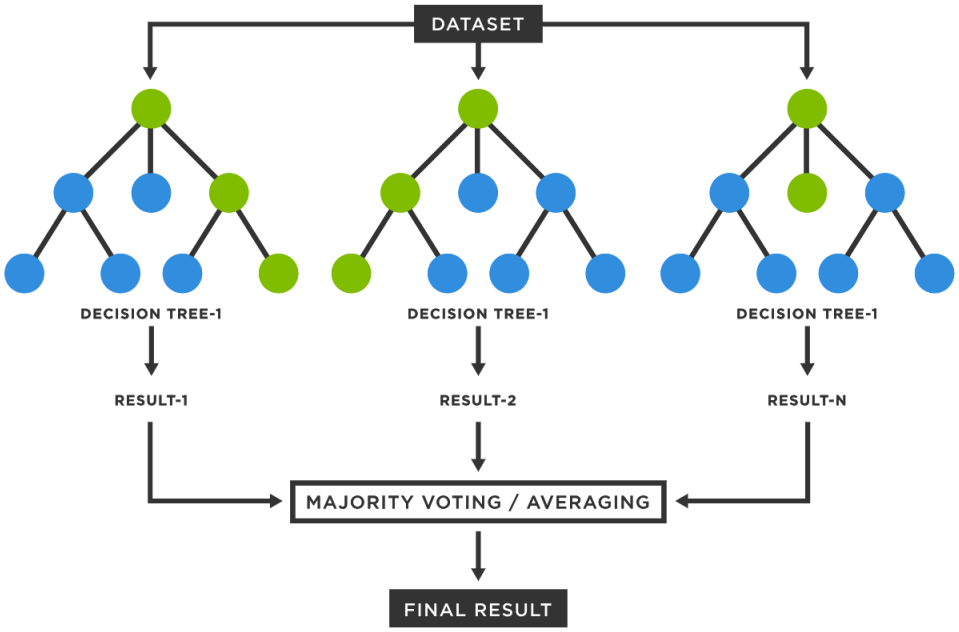

Figure 2.4: Random Forest example [37]

K-Nearest Neighbor

K-nearest neighbors (KNN) is a type of supervised learning algorithm used for both regression and classification. KNN tries to predict the correct class for the test data by calculating the distance between the test data and all the training points. Then select the K number of points which is closet to the test data. The KNN algorithm calculates the probability of the test data belonging to the classes of 'K' training data and class holds the highest probability will be selected. In the case of regression, the value is the mean of the 'K' selected training points. With the help of K-NN, we can easily identify the category or class of a particular dataset. KNN (K-Nearest Neighbors) involves determining, for each new individual that we want to classify, the list of the closest neighbors among the already classified individuals. The individual is assigned to the class that contains the most individuals among these closest neighbors. This method requires choosing a distance metric, with the most common one being the Euclidean distance, and the number of neighbors to consider. Selecting the value of K depends on the the count of the nearest neighbors. We have to compute distances between test points and trained labels points. Updating distance metrics with every iteration is computationally expensive, and that's why KNN is a lazy learning algorithm [38].

The figure 2.5 shows an example of KNN classification:

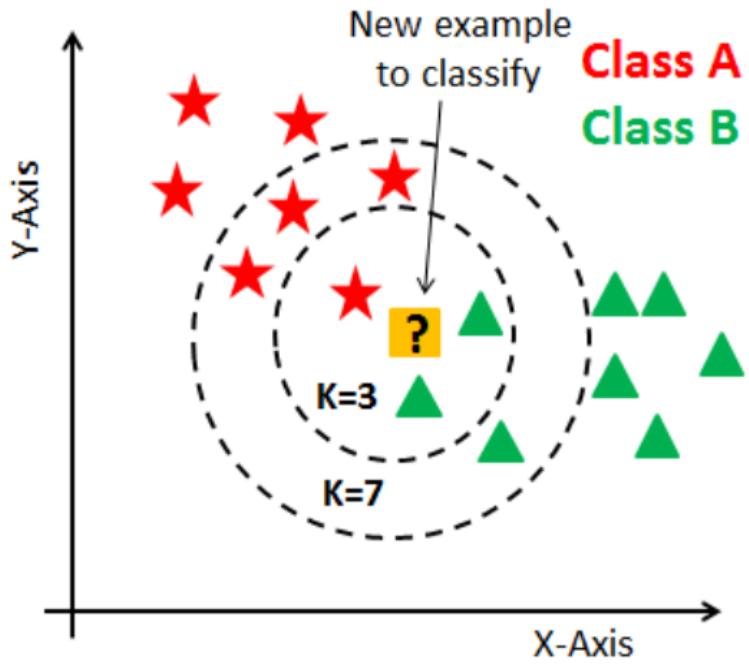

Figure 2.5: Example of KNN classification [39]

As it can be verified from the above image, if we proceed with K=3, then we predict that the test input belongs to class B. And if we continue with K=7, then we predict that test input belongs to class A. That's how you can imagine that the K value has a powerful effect on KNN performance. Actually, There are no pre-defined statistical methods to find the most favorable value of K, but choosing a small value of K leads to unstable decision boundaries.

Naive bayes

Naive Bayes classifiers encompass a group of classification algorithms rooted in Bayes' Theorem. Rather than being a single algorithm, it is a family of algorithms that adhere to a shared principle: the independence assumption. This assumption implies that each pair of features being classified is considered independent of one another. The Naïve Bayes algorithm is comprised of two words Naïve and Bayes, Which can be described as:

The term "Naïve" in Naïve Bayes stems from the assumption that the presence or occurrence of a specific feature is independent of the presence or occurrence of other features. For example, when identifying a fruit based on its color, shape, and taste, a red, spherical, and sweet fruit would be recognized as an apple. This implies that each

feature independently contributes to the identification of the fruit as an apple, without any dependence on the other features.The term ”Bayes” stands Bayes because it depends on the principle of Bayes' Theorem. [40]. The figure 2.6 shows an example of Naive Bayes classifier:

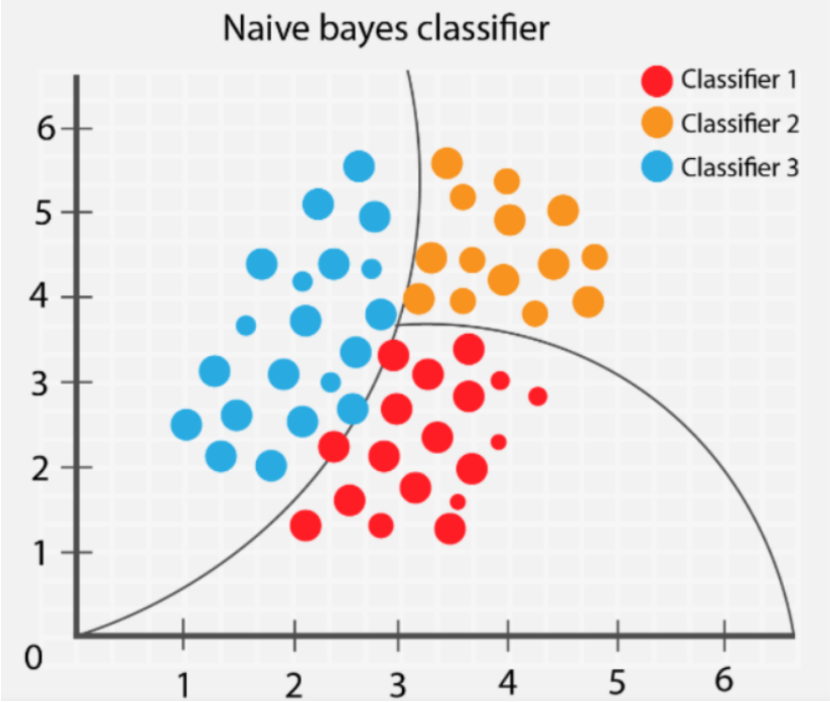

Figure 2.6: Naive Bayes Classifier [41]

Bayes' Theorem

Bayes' Theorem finds the probability of an event occurring given the probability of another event that has already occurred. Bayes' theorem is stated mathematically as the following equation:

$$P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)} \tag{2.3}$$

where: A and B are events and P(B)$a \neq b0$. Basically, we are trying to find probability of event A, given the event B is true. Event B is also termed as evidence. P(A) is the priori of A (the prior probability, i.e. Probability of event before evidence is seen). The evidence is an attribute value of an unknown instance(here, it is event B). P(A|B) is a posteriori probability of B, i.e. probability of event after evidence is seen [42].

2.3.2 Unsupervised learning

Unsupervised learning is the training of a machine using information that is neither classified nor labeled and allowing the algorithm to act on that information without guidance. Here the task of the machine is to group unsorted information according to similarities, patterns, and differences without any prior training of data. It allows the model to work on its own to discover patterns and information that was previously undetected. Basically it deals with the unlabelled data [43]. Unsupervised learning is classified into two categories of algorithms:

Clustering

Clustering is the unsupervised equivalent of classification. In clustering, the algorithm analyzes the data to find groups or clusters of data points that are similar to each other based on certain measures of similarity. Data points belonging to the same cluster are considered more similar to each other than to data points in other clusters[44].

The figure 2.7 shows an example of Clustering:

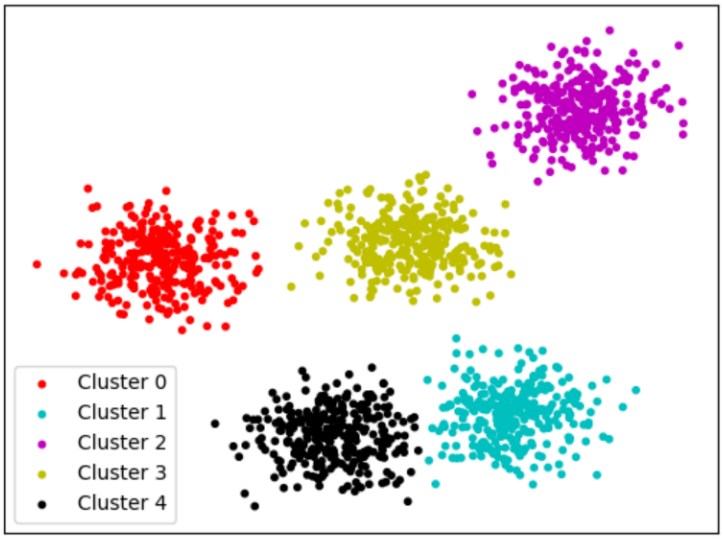

Figure 2.7: Example of clustering [45]

Association Rule Mining

This type of unsupervised machine learning takes a rule-based approach to discovering interesting relationships between features in a given dataset. It works by using a measure

of interest to identify strong rules found within a dataset. Association rules are ”if-then” statements, that help to show the probability of relationships between data items, within large data sets in various types of databases. Association rule mining has a number of applications and is widely used to help discover sales correlations in medical data sets [46].

2.3.3 Conclusion

In conclusion, in the second chapter, we provided a comprehensive overview of machine learning, its fundamental concepts, and various techniques used in the field. We explored the principles of supervised and unsupervised learning, understanding how models are trained to make predictions and discover patterns in data. Additionally, we delved into popular machine learning algorithms such as decision trees, support vector machines, and neural networks, highlighting their strengths and applications. Overall, this chapter laid the foundation for the subsequent chapter, where we will apply these machine learning techniques to tackle the challenge of intrusion detection in greater detail.

Chapter 3

Results and Discussion

3.1 Introduction

In this chapter, we provide a detailed account of the development process for our project, encompassing essential stages such as dataset creation, preprocessing, classification model construction, and the subsequent discussion of obtained results. We begin by describing the meticulous development of our dataset, including its composition, acquisition, and any necessary preprocessing steps. Next, we delve into the pre-processing phase, where we outline the techniques applied to cleanse, normalize, and enhance the dataset for optimal performance. Subsequently, we introduce our carefully crafted classification models, elaborating on the chosen algorithms, training methodologies, and model evaluation techniques. Finally, we present a comprehensive analysis and discussion of the results achieved, highlighting the performance, accuracy, and any notable observations or insights gained from our experiments.

3.2 System Architecture

Our system is a Network Intrusion Detection System (NIDS) that analyzes incoming packets using various machine learning techniques to classify them as either attack or normal, regardless of the type of attack. Our system is:

el **A single level system**: It has the ability to classify whether the incoming packets of our data are attacks or not.

The figure 3.1 shows the diagram of our intrusion detection system:

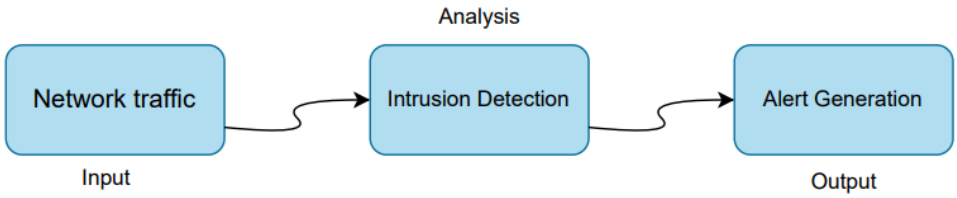

Figure 3.1: Intrusion Detection System Diagram

As we can see from the figure 3.1, at the core of our system, we find the input component, which consists of network traffic data, which is collected from different sources such as routers and switches. The analysis component forms the heart of our architecture, where machine learning algorithms are applied, the output component generates timely alerts, which provide a critical information about the nature detected intrusion.

3.3 Data Description

The KDD Cup 1999 dataset was created for the Knowledge Discovery and Data Mining (KDD) Cup competition held in 1999. It is a widely used dataset in the field of network intrusion detection and has become a benchmark dataset for evaluating intrusion detection systems. The dataset was derived from the 1998 DARPA Intrusion Detection Evaluation Program, which aimed to assess the effectiveness of intrusion detection systems in detecting various types of network attacks. It contains a large collection of network traffic data captured from a simulated environment that emulates a military network. The KDD Cup 1999 dataset consists of approximately 4 million connection records, including both normal and malicious network traffic instances. The connections are categorized into different attack types, such as denial of service (DoS), probing, user-to-root, and remote-to-local attacks. The NSL-KDD Cup 1999 dataset is an improved version of the original KDD Cup 1999 dataset. It was developed to address some limitations and issues present in the original dataset. The main motivation behind creating the NSL-KDD dataset was to address the problem of the highly imbalanced class distribution in the original dataset. The KDD Cup 1999 dataset suffered from an over-representation of certain

attack types, which made it challenging to develop effective intrusion detection models that could generalize well to real-world scenarios. The NSL-KDD dataset overcomes this issue by rebalancing the class distribution, ensuring a more even representation of different attack types. It also includes more diverse and representative instances, making it more suitable for evaluating the performance of intrusion detection systems across various attack scenarios [47].

Advantages of NSL-KDD over the original KDD data

- It does not include redundant records in the train set, so the classifiers will not be biased towards more frequent records.

- There is no duplicate records in the proposed test sets; therefore, the performance of the learners are not biased by the methods which have better detection rates on the frequent records.

- The number of selected records from each difficulty level group is inversely proportional to the percentage of records in the original KDD data set. As a result, the classification rates of distinct machine learning methods vary in a wider range, which makes it more efficient to have an accurate evaluation of different learning techniques.

- The number of records in the train and test sets are reasonable, which makes it affordable to run the experiments on the complete set without the need to randomly select a small portion. Consequently, evaluation results of different research works will be consistent and comparable [48].

Based on the official website, we will now present the following statistics:

Table 3.1 presents the statistics of redundant records in the KDD train set:

Table 3.1: Redundant records of the train test

	Reduction rate	**Original records**	**Distinct records**
Attacks	3,925,650	262,178	93.32 %
Normal	972,781	812,814	16.44 %
Total	4,898,431	1,074,992	78.05%

Table 3.2 presents the statistics of redundant records in the KDD test set:

Table 3.2: Redundant records of the test set

	Reduction rate	**Original records**	**Distinct records**
Attacks	250,436	29,378	88.26%
Normal	60,591	47,911	20.92%
Total	311,027	77,289	75.15%

It's important to note that similar to the KDD Cup 1999 dataset, the NSL KDD Cup 1999 dataset has undergone some preprocessing, including the removal of duplicate and redundant records. Additionally, the dataset has been transformed to ensure there are no missing values. This dataset offers a more balanced class distribution and a wider range of instances, making it suitable for evaluating intrusion detection systems in a more realistic and diverse setting [49].

3.4 Tools

Libraries

The workflow is based on the use of the Tensorflow library for building and training machine learning models. Numpy and pandas libraries for data manipulation and analysis, scikit-learn (sklearn) library that provides a wide range of tools and algorithms for model training, evaluation. The use of matplotlib and seaborn libraries for data visualization and exploration, they provide the creation of various types of plots, including line plots, scatter plots, bar plots, histograms, and more. The warnings library is part of Python's standard library and it provides a way to handle and control warning messages.

3.4.1 Python programming language

Python is an open-source, interpreted, object-oriented, and high-level programming language. Its design philosophy emphasizes code readability to reduce the cost of program maintenance. Besides, Python supports modules and packages which promote using it for several applications such as web development (server-side), software development, mathematics, and system scripting [50].

3.4.2 Google Colaboratory

Colab, developed by Google as we, is an innovative product that empowers users to write and run Python code directly in their web browser. It is particularly beneficial for tasks related to machine learning, data analysis, and education. Technically, Colab is a Jupyter notebook service hosted on the cloud, eliminating the need for any setup or installation. The best part is that it offers free access to computing resources, including GPUs, making it a valuable platform for various computational tasks [51].

3.5 Data Exploration

Noting that, we obtained our data set from Kaggle, a popular online platform for sharing and discovering data sets.

3.5.1 Libraries Importation

In the libraries importation section, we have imported the libraries previously mentioned, to facilitate the data exploration and analysis process. The algorithm details will be provided in the appendices section.

3.5.2 Feature Analysis

the data set consists of approximately 125,972 instances, 43 columns which includes 39 numerical attributes and 4 categorical attributes.

- Numerical attributes: A numerical attribute, also known as a quantitative attribute, is a type of variable that represents numeric values or measurements. Numerical attributes can be continuous like age, temperature, height, weight, and time duration.Continuous numerical attributes can have an infinite number of possible values within a range. Or discrete that can only take specific, separate values. Examples include the number of siblings, the number of items purchased, or the count of occurrences [52]. From our data set, we can find for example that the duration, src_bytes, dif_srv_rate are numerical attributes. Where src_bytes refers to the size of the data in bytes of a packet sent from the source host, and dif_srv_rate which refers to the rate of different services or protocols used in network connections.

- Categorical attributes: A categorical attribute is a type of variable that represents distinct categories or labels. Categorical attributes can take on a limited set of values, each representing a different category or class such as in our case 'Attack' or 'Normal' [53]. we can give an example related to our data set: the protocol_type, service, flag and outcome are categorical attributes.

The figures 3.2 and 3.3 show the data type of all the attributes:

```
RangeIndex: 125972 entries, 0 to 125971
Data columns (total 43 columns):
 #   Column                       Non-Null Count   Dtype
---  ------                       --------------   -----
 0   duration                     125972 non-null  int64
 1   protocol_type                125972 non-null  object
 2   service                      125972 non-null  object
 3   flag                         125972 non-null  object
 4   src_bytes                    125972 non-null  int64
 5   dst_bytes                    125972 non-null  int64
 6   land                         125972 non-null  int64
 7   wrong_fragment               125972 non-null  int64
 8   urgent                       125972 non-null  int64
 9   hot                          125972 non-null  int64
 10  num_failed_logins            125972 non-null  int64
 11  logged_in                    125972 non-null  int64
 12  num_compromised              125972 non-null  int64
 13  root_shell                   125972 non-null  int64
 14  su_attempted                 125972 non-null  int64
 15  num_root                     125972 non-null  int64
 16  num_file_creations           125972 non-null  int64
 17  num_shells                   125972 non-null  int64
 18  num_access_files             125972 non-null  int64
 19  num_outbound_cmds            125972 non-null  int64
```

Figure 3.2: Feature analysis

```
20  is_host_login                125972 non-null  int64
21  is_guest_login               125972 non-null  int64
22  count                        125972 non-null  int64
23  srv_count                    125972 non-null  int64
24  serror_rate                  125972 non-null  float64
25  srv_serror_rate              125972 non-null  float64
26  rerror_rate                  125972 non-null  float64
27  srv_rerror_rate              125972 non-null  float64
28  same_srv_rate                125972 non-null  float64
29  diff_srv_rate                125972 non-null  float64
30  srv_diff_host_rate           125972 non-null  float64
31  dst_host_count               125972 non-null  int64
32  dst_host_srv_count           125972 non-null  int64
33  dst_host_same_srv_rate       125972 non-null  float64
34  dst_host_diff_srv_rate       125972 non-null  float64
35  dst_host_same_src_port_rate  125972 non-null  float64
36  dst_host_srv_diff_host_rate  125972 non-null  float64
37  dst_host_serror_rate         125972 non-null  float64
38  dst_host_srv_serror_rate     125972 non-null  float64
39  dst_host_rerror_rate         125972 non-null  float64
40  dst_host_srv_rerror_rate     125972 non-null  float64
41  outcome                      125972 non-null  object
42  level                        125972 non-null  int64
dtypes: float64(15), int64(24), object(4)
```

Figure 3.3: Data type 2

3.5.3 Descriptive Statistics

Descriptive statistics are numerical measures that summarize and describe the basic characteristics of the dataset, it provides a summary of the dataset's key characteristics, allowing for a better understanding of the data's distribution, central tendency, and variability. Here is a description of the common descriptive statistics:

The mean: The mean, also known as the average, is the sum of all values in a dataset divided by the total number of values.

The standard deviation: The standard deviation measures the dispersion or spread of the values around the mean. It quantifies how much the values deviate from the average.

The minimum: The minimum is the smallest value observed in a dataset. It represents the lowest point or boundary of the data.

The 25th Percentile (Q1): The 25th percentile, also known as the lower quartile or first quartile, is the value below which 25% of the data points fall and it divides the data into four equal parts.

The 50th Percentile (Q2 or Median): The 50th percentile, also known as the median, is the value that divides the data into two equal halves. It represents the middle value when the data is sorted in ascending order.

75th Percentile (Q3): The 75th percentile, also known as the upper quartile or third quartile, is the value below which 75% of the data points fall [54].

The figures from 3.4 to 3.8 show our dataset descriptive analysis:

	duration	src_bytes	dst_bytes	land	wrong_fragment	urgent	hot	num_failed_logins
count	125972.000000	125972.000000	125972.000000	125972.000000	125972.000000	125972.000000	125972.000000	125972.000000
mean	287.146929	45567.100824	19779.271433	0.000198	0.022688	0.000111	0.204411	0.001222
std	2604.525522	5870354.480801	4021285.112114	0.014086	0.253531	0.014366	2.149977	0.045239
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
50%	0.000000	44.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
75%	0.000000	276.000000	516.000000	0.000000	0.000000	0.000000	0.000000	0.000000
max	42908.000000	1379963888.000000	1309937401.000000	1.000000	3.000000	3.000000	77.000000	5.000000

Figure 3.4: Descriptive statistics 1

logged_in	num_compromised	root_shell	su_attempted	num_root	num_file_creations	num_shells	num_access_files	num_outbound_cmds
125972.000000	125972.000000	125972.000000	125972.000000	125972.000000	125972.000000	125972.000000	125972.000000	125972.000000
0.395739	0.279253	0.001342	0.001103	0.302194	0.012669	0.000413	0.004096	0.000000
0.489011	23.942137	0.036603	0.045155	24.399715	0.483937	0.022181	0.099370	0.000000
0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
1.000000	7479.000000	1.000000	2.000000	7468.000000	43.000000	2.000000	9.000000	0.000000

Figure 3.5: Descriptive statistics 2

is_host_login	is_guest_login	count	srv_count	serror_rate	srv_serror_rate	rerror_rate	srv_rerror_rate	same_srv_rate
125972.000000	125972.000000	125972.000000	125972.000000	125972.000000	125972.000000	125972.000000	125972.000000	125972.000000
0.000008	0.009423	84.108207	27.738093	0.284487	0.282488	0.119959	0.121184	0.660925
0.002817	0.096613	114.508828	72.636092	0.446457	0.447024	0.320437	0.323648	0.439624
0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
0.000000	0.000000	2.000000	2.000000	0.000000	0.000000	0.000000	0.000000	0.090000
0.000000	0.000000	14.000000	8.000000	0.000000	0.000000	0.000000	0.000000	1.000000
0.000000	0.000000	143.000000	18.000000	1.000000	1.000000	0.000000	0.000000	1.000000
1.000000	1.000000	511.000000	511.000000	1.000000	1.000000	1.000000	1.000000	1.000000

Figure 3.6: Descriptive statistics 3

diff_srv_rate	srv_diff_host_rate	dst_host_count	dst_host_srv_count	dst_host_same_srv_rate	dst_host_diff_srv_rate	dst_host_same_src_port_rate
125972.000000	125972.000000	125972.000000	125972.000000	125972.000000	125972.000000	125972.000000
0.063053	0.097322	182.149200	115.653725	0.521244	0.082952	0.148379
0.180315	0.259831	99.206565	110.702886	0.448950	0.188922	0.308998
0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
0.000000	0.000000	82.000000	10.000000	0.050000	0.000000	0.000000
0.000000	0.000000	255.000000	63.000000	0.510000	0.020000	0.000000
0.060000	0.000000	255.000000	255.000000	1.000000	0.070000	0.060000
1.000000	1.000000	255.000000	255.000000	1.000000	1.000000	1.000000

Figure 3.7: Descriptive statistics 4

dst_host_srv_diff_host_rate	dst_host_serror_rate	dst_host_srv_serror_rate	dst_host_rerror_rate	dst_host_srv_rerror_rate	level
125972.000000	125972.000000	125972.000000	125972.000000	125972.000000	125972.000000
0.032543	0.284455	0.278487	0.118832	0.120241	19.504056
0.112564	0.444785	0.445670	0.306559	0.319460	2.291512
0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
0.000000	0.000000	0.000000	0.000000	0.000000	18.000000
0.000000	0.000000	0.000000	0.000000	0.000000	20.000000
0.020000	1.000000	1.000000	0.000000	0.000000	21.000000
1.000000	1.000000	1.000000	1.000000	1.000000	21.000000

Figure 3.8: Descriptive statistics 5

3.5.4 Data Balance

A balanced dataset refers to having an equal or nearly equal number of instances for each class or categorym, this ensures that there is no significant class imbalance that could bias the model's performance. A balanced dataset allows the machine learning models to learn and make predictions for each class more effectively. It ensures that the model receives sufficient examples from each class, reducing the risk of bias towards the majority class. Having a balanced distribution is particularly important in the field of attack detection because it helps ensure that your machine learning models are exposed to a sufficient number of instances from each class. This balance allows the models to learn the patterns and characteristics of both attacks and normal instances, leading to more accurate and reliable predictions. I have conducted a test to evaluate the balance of the dataset, specifically focusing on the two categories of interest: 'Protocole_type' and 'outcome' by calculating the number of instances of each category.

The result of the test is shown in the figure 3.9:

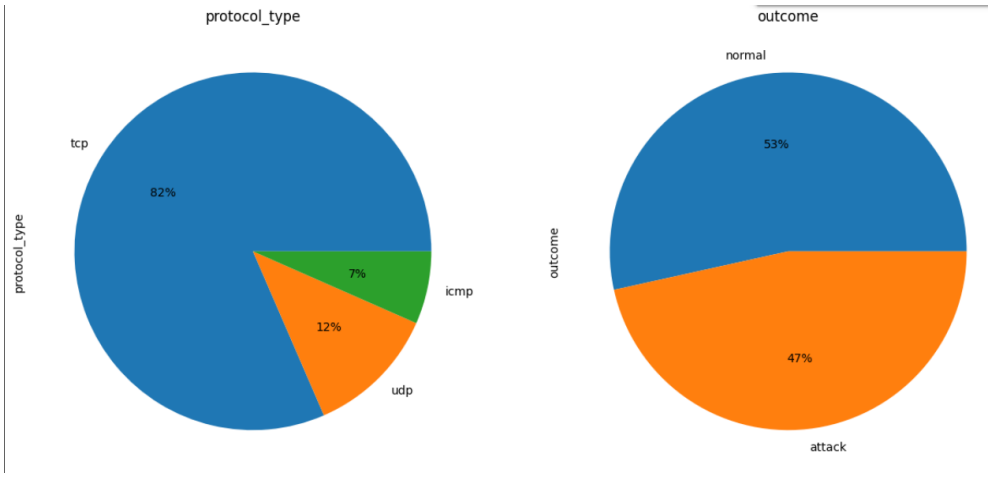

Figure 3.9: Balanced Data test

Discussion: A balanced distribution in the ”outcome” category, with 47% representing attacks and 53% representing normal instances, suggests that my dataset is fairly representative of both classes and in the context of your project, where the goal is to detect attacks, having a balanced data in the class of outcome is a positive development. By achieving a balanced distribution in the ”outcome” category, the dataset enables the models to learn and generalize well to both attack and normal instances, reducing the risk of biased predictions and improving the overall performance and robustness of the system.

3.6 Data Preprocessing

In the preprocessing phase, I applied various steps to prepare the dataset for further analysis. This included scaling the numerical columns, performing one-hot encoding on selected categorical columns and applying principle common technique.

3.6.1 Dummy Encoding

Dummy encoding, also known as ”one-hot encoding”, is a popular technique in data preprocessing that transforms categorical variables into numerical representations suitable for machine learning algorithms. In Dummy encoding method, a new binary column is created for each category, representing the presence (1) or absence (0) of that category in

a specific observation. If a row belongs to a particular category, the corresponding column will have a value of 1, and all other columns will have a value of 0. As a result, the number of columns increased from 43 to 124, reflecting the creation of additional binary columns through one-hot encoding.

The figures 3.10 and 3.11 show the results of Dummy encoding method:

	outcome	land	logged_in	is_host_login	is_gues
0	0	0.0	0.0	0.0	
1	1	0.0	0.0	0.0	
2	0	0.0	1.0	0.0	
3	0	0.0	1.0	0.0	
4	1	0.0	0.0	0.0	
...	...	...	...	...	
125967	1	0.0	0.0	0.0	
125968	0	0.0	0.0	0.0	
125969	0	0.0	1.0	0.0	
125970	1	0.0	0.0	0.0	
125971	0	0.0	1.0	0.0	

125972 rows × 124 columns

Figure 3.10: Dummy encoding result 1

flag_S1	flag_S2	flag_S3	flag_SF	flag_SH
0	0	0	1	0
0	0	0	0	0
0	0	0	1	0
0	0	0	1	0
0	0	0	0	0
...	...	...	...	...
0	0	0	0	0
0	0	0	1	0
0	0	0	1	0
0	0	0	0	0
0	0	0	1	0

Figure 3.11: Dummy encoding result 2

3.7 Modeling

In the context of your project, modeling refers to the process of building and training machine learning models to make predictions from our intrusion detection data set. It involves selecting appropriate models, configuring them, and training them using your labeled data. The purpose of modeling is to leverage the power of these algorithms to detect and classify attacks accurately.

3.7.1 Confusion matrix

A confusion matrix is a square matrix that summarizes the performance of a classification model by displaying the counts of true positive, true negative, false positive, and false negative predictions. It allows us to analyze the accuracy and error rates of our model for each class or category in our dataset. In the context of intrusion detection, the confusion matrix can help us to know how well our model is identifying attacks and normal instances. It provides detailed information on the following metrics:

- True Positives (TP): The number of correctly predicted positive instances, indicating correctly identified attacks.

- True Negative (TN): The number of correctly predicted negative instances, indicating correctly identified normal instances.

- False Positives (FP): The number of instances that were incorrectly predicted as positive (attacks) but are actually negative (normal instances).

- False Negatives (FN): The number of instances that were incorrectly predicted as negative (normal instances) but are actually positive (attacks).

The figure 3.12 demonstrates a confusion matrix:

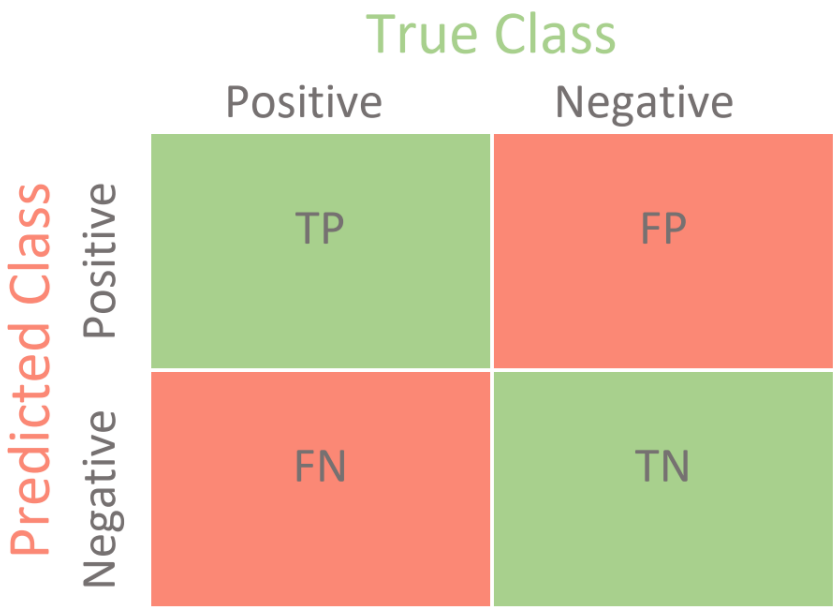

Figure 3.12: Binary classification confusion matrix

Using these metrics, we can calculate various performance indicators such as accuracy, precision and recall which provide a comprehensive assessment of our model's performance in terms of both detecting attacks and correctly identifying normal instances.

Accuracy:

It measures the overall correctness of the model's predictions and it is calculated using the following formula:

$$Accuracy = (TP + TN)/(TP + TN + FP + FN). \tag{3.1}$$

Precession:

It measures the proportion of correctly predicted positive instances (attacks) out of all instances predicted as positive. It is calculated using the following formula:

$$Precession = TP/(TP + FP) \tag{3.2}$$

Precision indicates how precise the model is in identifying attacks, and a higher precision value indicates fewer false positives.

Recall:

Recall (also known as sensitivity or true positive rate): It measures the proportion of correctly predicted positive instances (attacks) out of all actual positive instances. It is calculated using the following formula:

$$Recall = TP/(TP + FN) \tag{3.3}$$

Recall indicates how well the model captures attacks, and a higher recall value indicates fewer false negatives.

By analyzing these metrics and the corresponding values in the confusion matrix, we can evaluate the performance of our intrusion detection models and make informed decisions about their effectiveness in identifying attacks and normal instances.

we also evaluate the performance of our intrusion detection system using various performance metrics and visualizations. Before diving into the results, it is important to define some key terms and metrics that will be discussed. These metrics provide insights into the model's ability to distinguish between normal and attack instances.

ROC Curve (Receiver Operating Characteristic Curve):

The ROC curve is a graphical representation of the trade-off between the true positive rate (TPR) and the false positive rate (FPR) at various classification thresholds. It helps us assess the model's performance across different threshold values.

Threshold:

The threshold is a value used to classify instances as either normal or attack based on their predicted probabilities. By adjusting the threshold, we can control the balance between false positives and false negatives in our predictions.

False Positive Rate (FPR):

The FPR is the proportion of instances predicted as attacks that are actually normal. It is calculated using the following formula:

$$FPR = FP/(FP + TN) \tag{3.4}$$

True Positive Rate (TPR):

The TPR is the proportion of actual attack instances that are correctly classified as attacks by the model. It is calculated using the following formula:

$$TPR = TP/(TP + FN) \tag{3.5}$$

3.7.2 Area under curve (AUC):

The Area Under the Curve (AUC) is a commonly used metric to quantify the performance of the ROC curve. It represents the probability that a randomly chosen positive instance will be ranked higher than a randomly chosen negative instance by the model. AUC ranges from 0 to 1, where a value of 0.5 indicates a random classifier and a value of 1 indicates a perfect classifier. A higher AUC value signifies a better discriminative ability of the model in distinguishing between positive and negative instances.

3.8 Results and Discussion

In this section, we present the results and discussion of our analysis using various machine learning models for intrusion detection. Throughout the previous chapter, we explored and described the implementation of different models, including Logistic Regression, K-Nearest Neighbors (KNN), Naive Bayes, Support Vector Machine and Artificial Neural Networks (ANN). These models were carefully chosen based on their potential to accurately classify instances as attacks or normal. The objective of this analysis was to evaluate the performance of these models and compare their effectiveness in detecting and classifying attacks. To assess their performance, we utilized well-established evaluation

criteria such as accuracy, precision and recall. The following subsections will provide a comprehensive analysis of the results, focusing on each model individually. This will enable us to gain a deeper understanding of their performance and contribute to the ongoing as we in the field of intrusion detection.

3.8.1 Logistic Regression

In order to evaluate the performance of the logistic regression model, we will present the Receiver Operating Characteristic (ROC) curve in the figure 3.13:

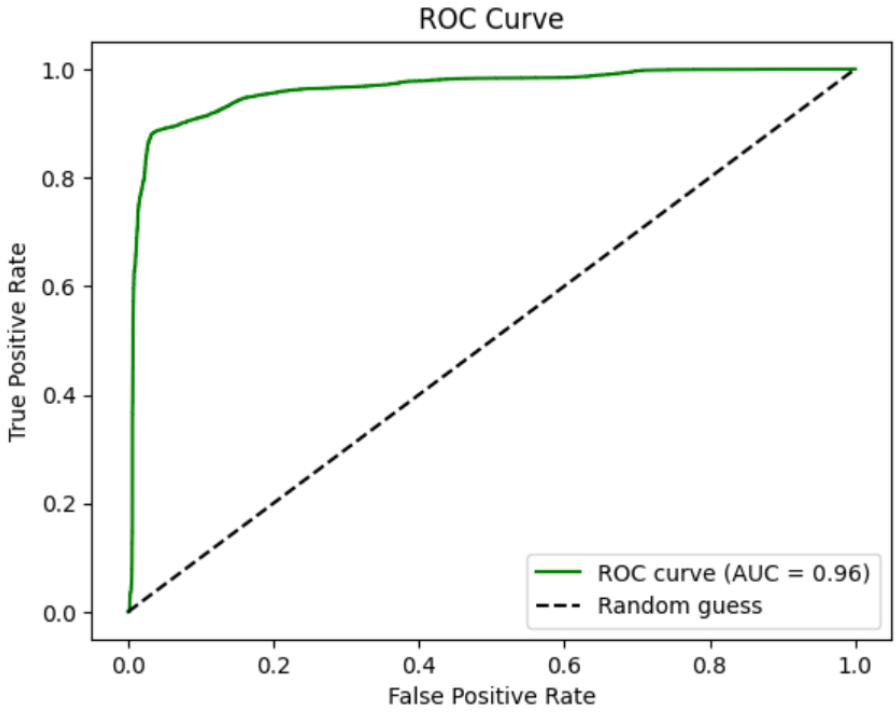

Figure 3.13: The ROC curve for Logistic Regression model

Discussion: The figure 3.13 shows the ROC curve of Logistic Regression model, it illustrates its performance across various classification thresholds. At low false positive rates FPR=0, the model achieves a reasonably high true positive rate TPR $\approx$ 0.85, indicating its ability to accurately classify positive instances while minimizing false positives. As the false positive rate increases (FPR=0.2), the TPR rises to around 0.9, demonstrating that the model maintains its effectiveness in correctly classifying positives despite a slightly higher rate of false positives. Notably, when the false positive rate reaches 0.4, 0.6, 0.8 and 1.0, the TPR stabilizes at 1, indicating perfect detection of positive instances, although it may result in a higher rate of false positives. Overall, the ROC curve show-

cases the logistic regression model's strong discriminatory power and its ability to strike a balance between true positive and false positive rates across different thresholds. With an AUC value of 0.96, the logistic regression model demonstrates a robust ability to differentiate between positive and negative instances. It suggests that the model performs well in ranking positive instances higher than negative instances, achieving accurate predictions in the majority of cases.

As mentioned earlier, the confusion matrix provides a comprehensive overview of the classification model's performance. In the figure 3.14, we present the confusion matrix to further analyze the results. It consists of the counts of true positives, true negatives, false positives, and false negatives, allowing us to evaluate the accuracy and effectiveness of the model in predicting the class labels.

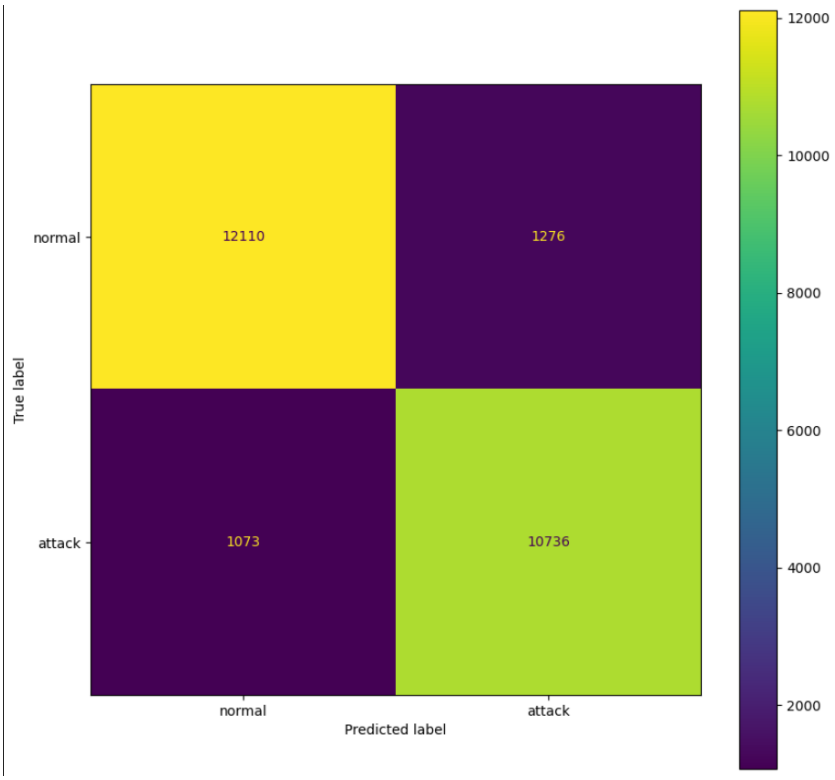

Figure 3.14: The confusion matrix of Logistic Regression model

The resulted evaluations metrics are given in Table 3.1:

Table 3.3: Classification report of logistic regression model

	Accuracy	**Precision**	**Recall**
Training	90.98	89.70	91.04
Testing	90.67	89.37	90.91

3.8.2 k-nearest neighbors(KNN)

In order to evaluate the performance of K-nearest neighbors model, we will present the Receiver Operating Characteristic (ROC) curve in the figure 3.15:

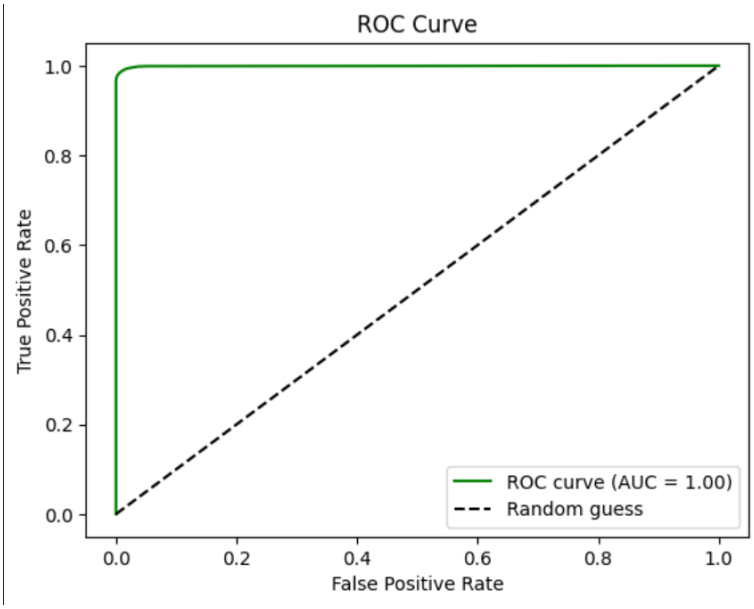

Figure 3.15: The ROC curve of KNN model

Discussion: From the figure 3.15 that demonstrates the ROC curve of KNN model, we can assume that it reveals an interesting pattern. When the false positive rate (FPR) is at its lowest value, i.e, FPR=0, the true positive rate (TPR) is equal to 1. This implies that the model achieves perfect detection of positive instances without any false positives. It suggests that the KNN model performs exceptionally well at this threshold, providing a high level of accuracy in identifying positive cases. As the false positive rate gradually increases to FPR=0.2 and beyond, the true positive rate remains consistently at 1. This signifies that the KNN model continues to maintain a perfect detection rate of positive instances, even in the presence of false positives. Remarkably, as the FPR equals to 0.4, 0.6, 0.8 until it reaches its highest value, i.e FPR=1.0, the true positive rate still remains at

1. This indicates that the KNN model classifies all positive instances correctly, regardless of the increased rate of false positives. In summary, the ROC curve of my KNN model exhibits a remarkable performance, with a perfect true positive rate across various false positive rates. This suggests that the model is adept at distinguishing positive instances and maintaining a high level of accuracy, even in the presence of false positives. The Area Under the Curve (AUC) represents the model's ability to discriminate between positive and negative instances, with a value of 1 indicating that the model perfectly separates the two classes without any misclassifications.

We present in the figure 3.16 the confusion matrix of KNN model:

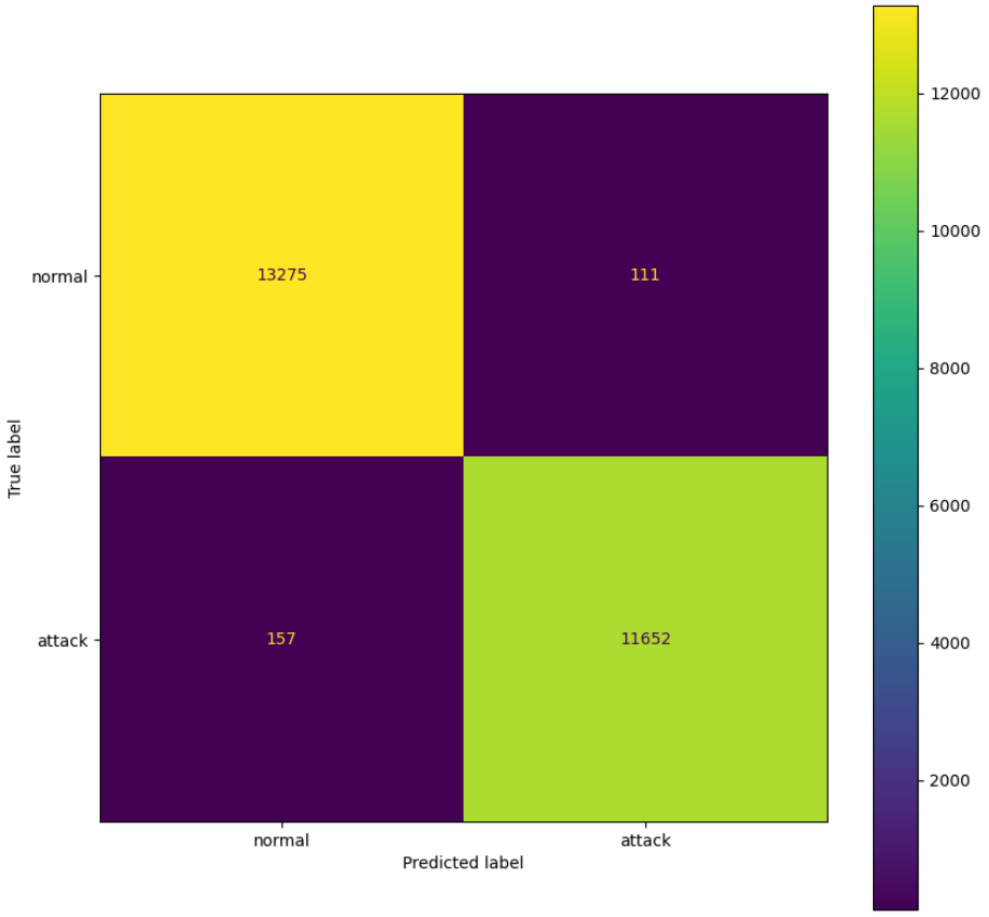

Figure 3.16: The confusion matrix of KNN model

The resulted evaluations metrics are given in Table 3.2:

Table 3.4: Classification report of KNN model

	Accuracy	**Precision**	**Recall**
Training	99.05	99.22	98.73
Testing	98.93	99.05	98.67

3.8.3 Naive Bayes

In order to evaluate the performance of Naive Bayes model, we will present the Receiver Operating Characteristic (ROC) curve in the figure 3.17:

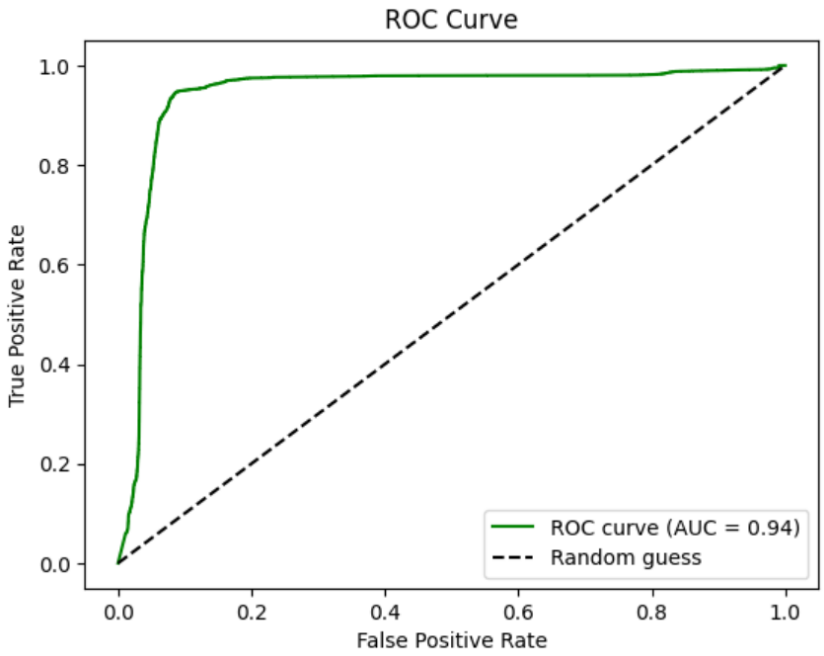

Figure 3.17: The ROC curve of Naive Bayes

Discussion: from the figure 3.17 that illustrates the ROC curve of Naive Bayes model, we can deduce that the model acheives initially a high true positive rate while keeping the false positive rate relatively low. However, as the false positive rate increases, the model's true positive rate reaches a stable value of 1, indicating perfect detection of positive instances. An AUC value of 0.94 for the Naive Bayes model's ROC curve indicates a strong performance in terms of classification accuracy, it suggests that the Naive Bayes model has a high probability of correctly ranking a randomly selected positive instance higher than a randomly selected negative instance.

We present in the figure 3.18 the confusion matrix of Naive Bayes model:

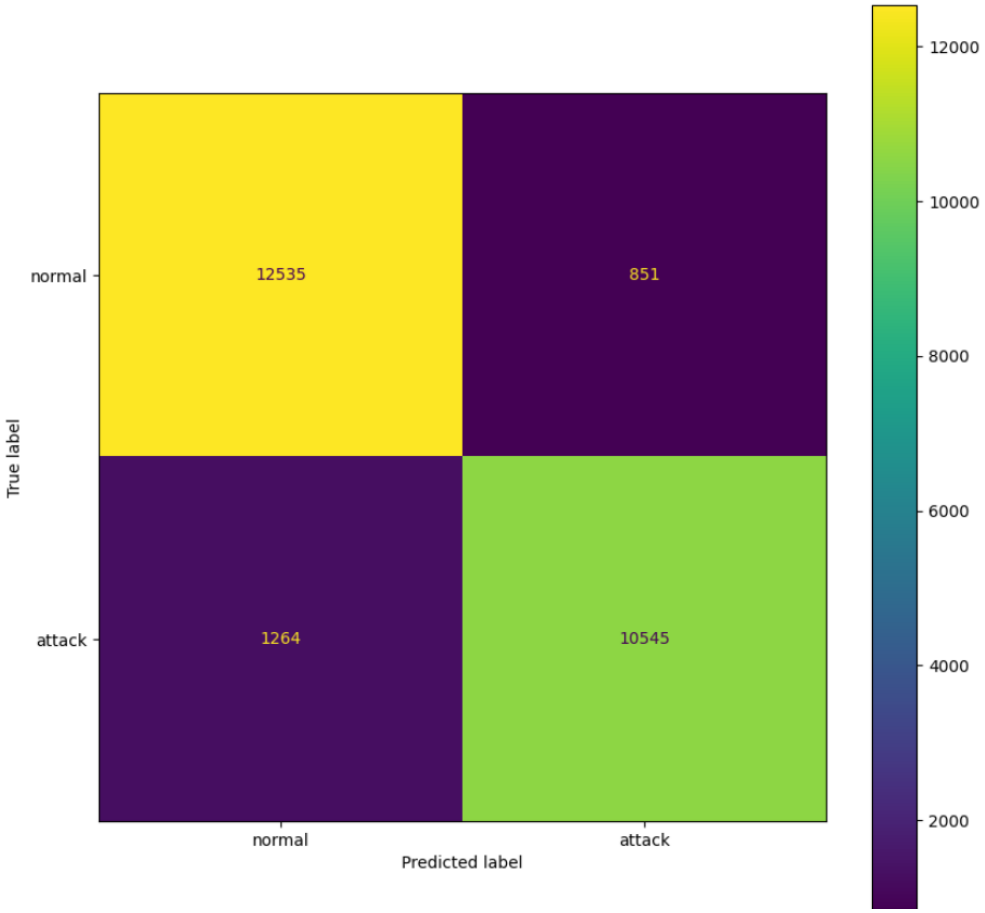

Figure 3.18: The confusion matrix of Naive Bayes model

The resulted evaluations metrics are given in Table 3.3:

Table 3.5: Classification report of Naive Bayes model

	Accuracy	**Precision**	**Recall**
Training	91.80	92.62	89.47
Testing	91.60	92.53	89.29

3.8.4 Support Vector Machines

To assess the performance of Support Vector Machines model, we will showcase the Receiver Operating Characteristic (ROC) curve in the figure 3.19.

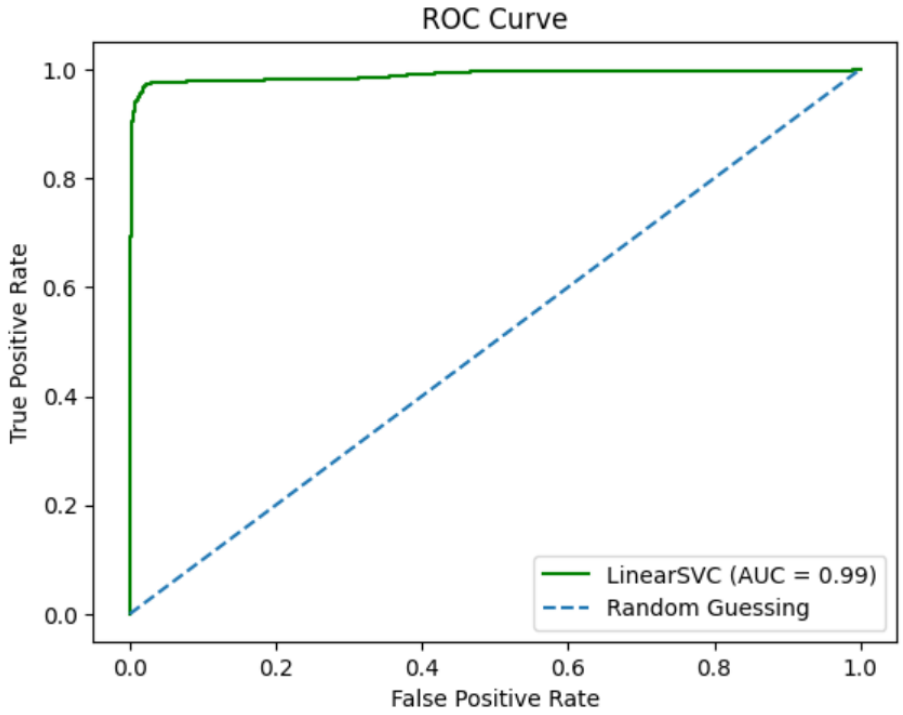

Figure 3.19: The ROC curve of Support Vector Machines

Discussion: From the figure 3.19 that illustrates the ROC curve of Support Vector Machines, we can deduce that the model its exceptional performance. When the false positive rate (FPR) is at its lowest value of 0, the true positive rate (TPR) is approximately 1, This indicates that the model achieves a perfect detection rate for positive instances. The TPR stabilizes at 1 when the FPR reaches 0.6, this suggests that, despite the higher rate of false positives, the SVM model maintains its ability to identify positive instances accurately and consistently. The Area Under the Curve (AUC) value of 0.99 for our SVM model's ROC curve indicates excellent overall performance. AUC represents the probability that the model will rank a randomly chosen positive instance higher than a randomly chosen negative instance. A value of 0.99 suggests that the SVM model has a high discriminatory power, with a strong ability to separate positive and negative instances.

Find in the figure 3.20 the confusion matrix of Support Vector Machine model:

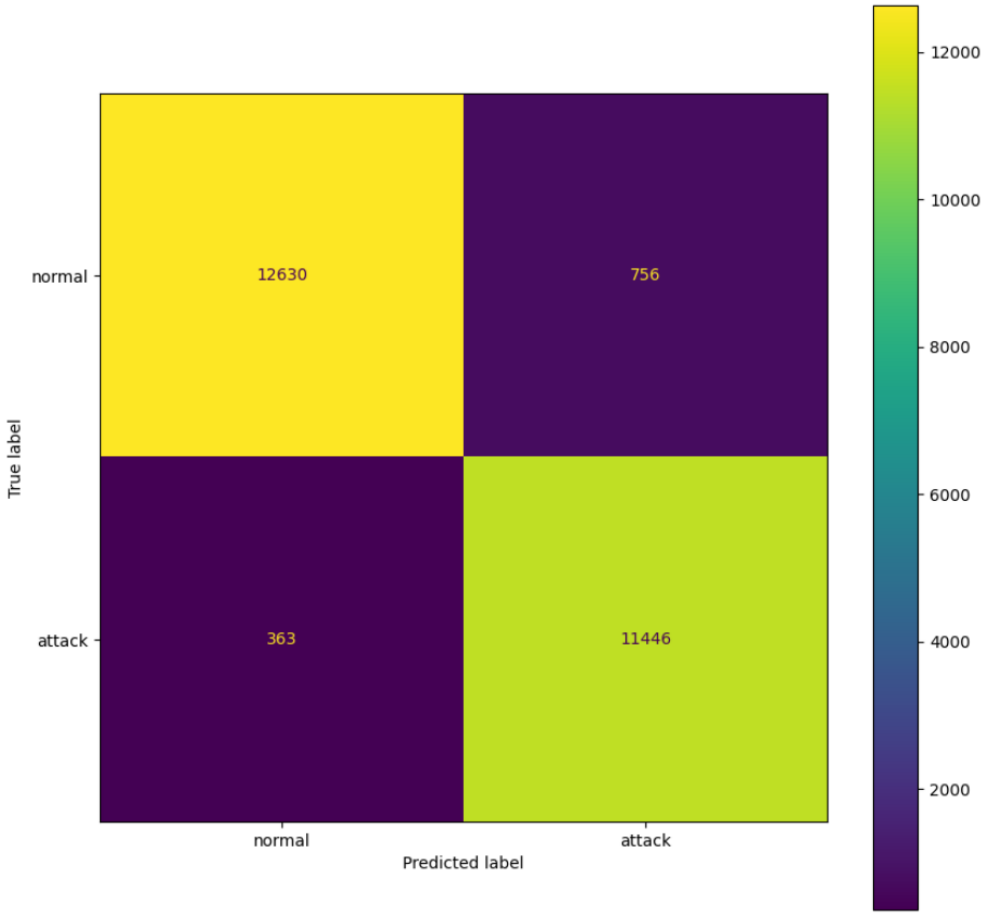

Figure 3.20: The confusion matrix of Support Vector Machines model

The resulted evaluations metrics are given in Table 3.4:

Table 3.6: Classification report of Support Vector Machine model

	Accuracy	**Precision**	**Recall**
Training	95.86	94.12	97.08
Testing	95.55	93.80	96.92

3.8.5 Decision Tree

Find in the figure 3.21, the ROC curve of the Decision Tree model:

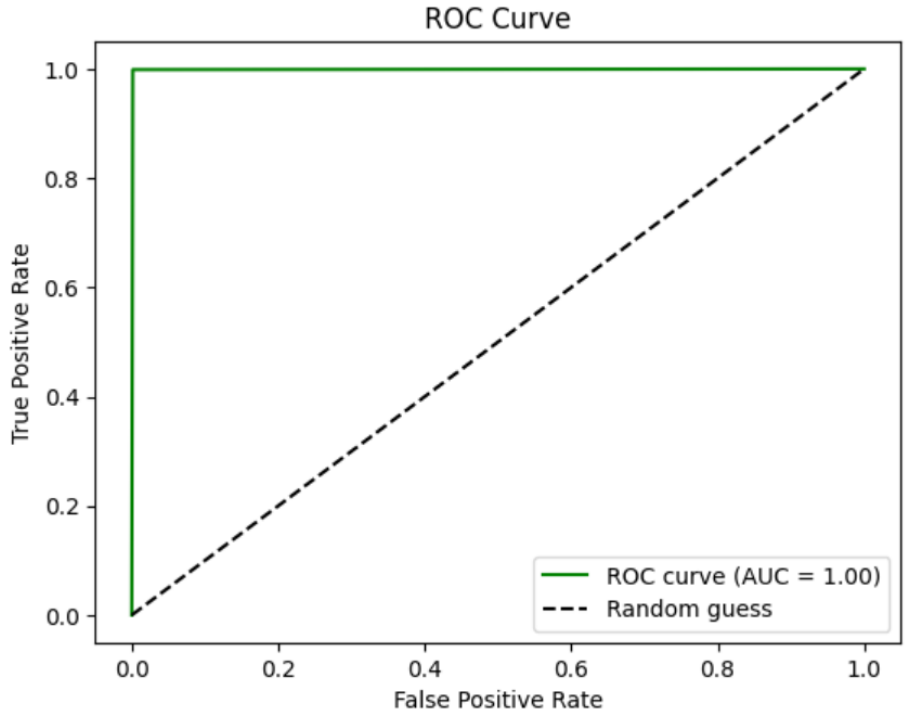

Figure 3.21: The ROC curve of Decision Tree

Discussion: From the figure 3.21 that demonstrates the ROC curve of Decision Tree model, it displays an excellent performance. When the false positive rate (FPR) is at any value from 0.0 to 1.0 (specifically, FPR=0.0, 0.2, 0.4, 0.6, 0.8, and 1.0), the true positive rate (TPR) consistently reaches 1.0. This indicates that the model achieves perfect detection of positive instances across all thresholds, regardless of the presence of false positives. Furthermore, the Area Under the Curve (AUC) value of 1.00 reinforces the model's outstanding performance, the value of 1.00 indicates that the model achieves perfect classification, ranking all positive instances higher than negative instances across all possible thresholds.

We present in the figure 3.22 the confusion matrix of Decision Tree model:

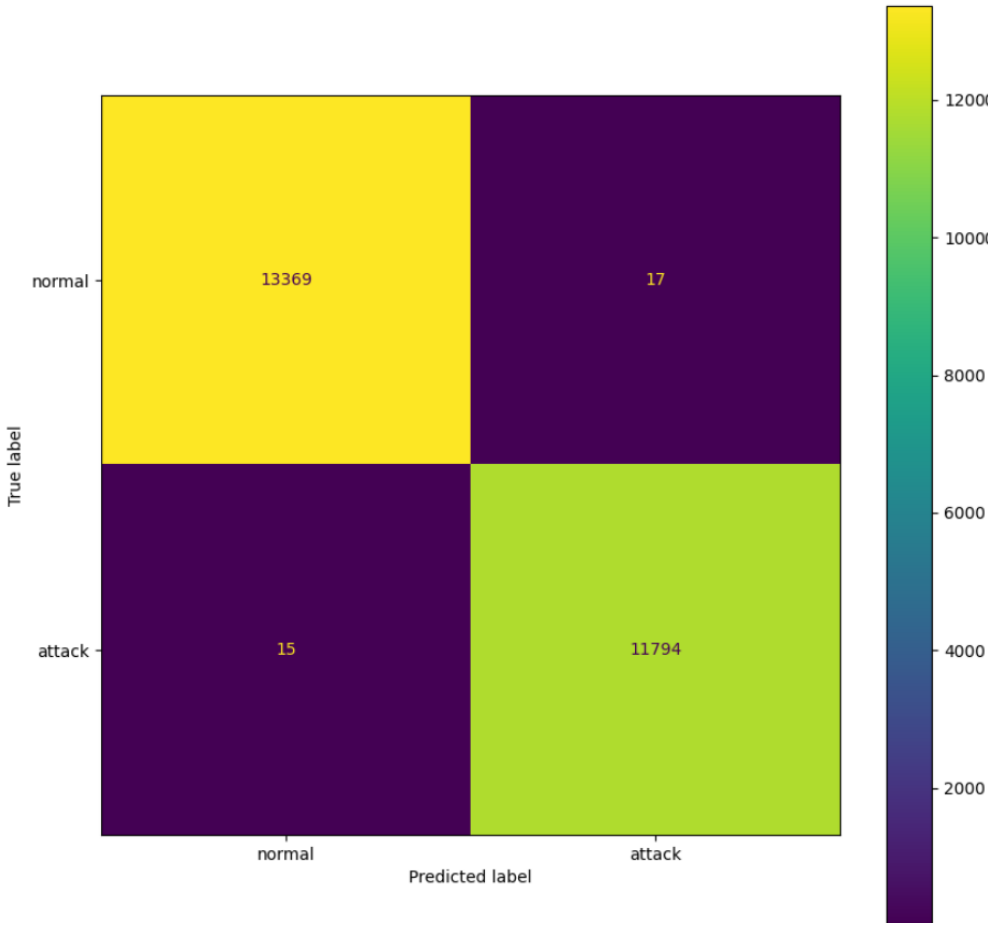

Figure 3.22: The confusion matrix of Decision Tree model

The resulted evaluations metrics are given in Table 3.5:

Table 3.7: Classification report of Decision Tree model

	Accuracy	**Precision**	**Recall**
Training	99.99	100.0	99.98
Testing	99.97	99.85	99.87

3.8.6 Random Forest

we will visualize the performance of Random Forest model using the Receiver Operating Characteristic (ROC) curve displayed in the figure 3.23:

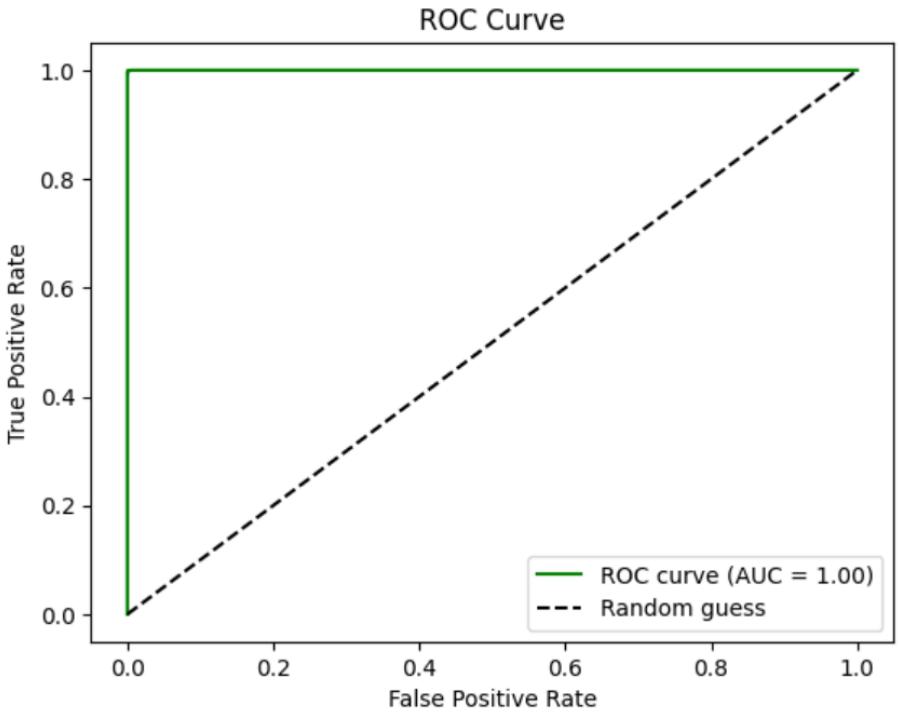

Figure 3.23: The ROC curve of Random Forest

Dicussion: From the figure 3.23 that demonstrates the ROC curve of Random Forest model, it displays an excellent performance. When the false positive rate (FPR) is at any value from 0.0 to 1.0 (specifically, FPR=0.0, 0.2, 0.4, 0.6, 0.8, and 1.0), the true positive rate (TPR) consistently reaches 1.0. This indicates that the model achieves perfect detection of positive instances across all thresholds, regardless of the presence of false positives. Furthermore, the Area Under the Curve (AUC) value of 1.00 reinforces the model's outstanding performance, the value of 1.00 indicates that the model achieves perfect classification, ranking all positive instances higher than negative instances across all possible thresholds.

We present in the figure 3.24 the confusion matrix of Random Forest model:

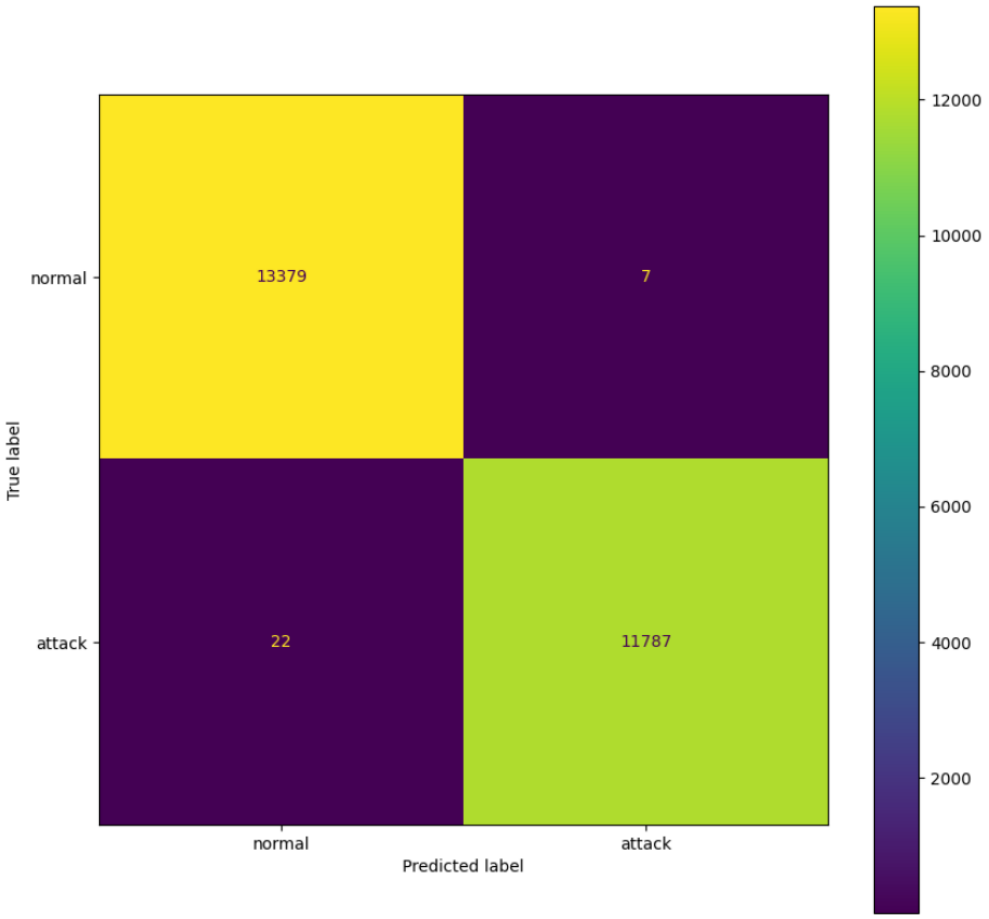

Figure 3.24: The confusion matrix of Random Forest model

The resulted evaluations metrics are given in Table 3.6:

Table 3.8: Classification report of Random Forest model

	Accuracy	**Precision**	**Recall**
Training	99.99	99.99	99.99
Testing	99.88	99.94	99.81

3.9 Results Comparison with recent works

In the figure 3.25, we present the graph that displays both training and testing accuracy of each model:

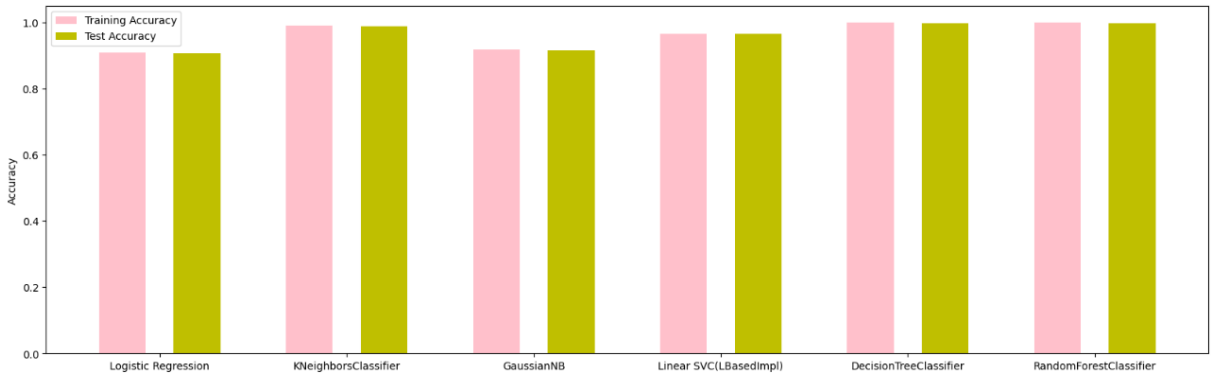

Figure 3.25: Training and testing accuracy of each model

Discussion: From the figure 3.25 that demonstrated the training and testing accuracy of each model, we can deduce that the decision tree classifier and the random forest classifier achieved a maximum accuracy value of 1.0. This indicates that these models were able to perfectly classify the training data. The fact that both the decision tree classifier and the random forest classifier achieved the same high accuracy on both the training and testing data sets suggests that overfitting may not be a significant concern in this case, in fact, it indicates a balanced and reliable performance. While the decision tree classifier and random forest classifier achieve a maximum accuracy of 1, it is important to recognize the strong performance of the logistic regression, KNN, Naive bayes and SVM models in terms of accuracy. Although they may not reach the maximum value, these models consistently achieve a significantly high accuracy, indicating their proficiency in correctly classifying instances. The KNN model, leveraging the concept of nearest neighbors, demonstrates robust performance in accurately assigning class labels. Similarly, logistic regression, utilizing a probabilistic framework, achieves a commendable accuracy by effectively capturing the relationship between the features and the target variable. Additionally, the SVM model, with its ability to find optimal decision boundaries, showcases notable accuracy performance. Collectively, these models contribute to the diverse range of approaches available for achieving accurate classification results and highlight their respective strengths in handling the given dataset.

In the figure 3.26, we present the graph that displays both training and testing precision of each model:

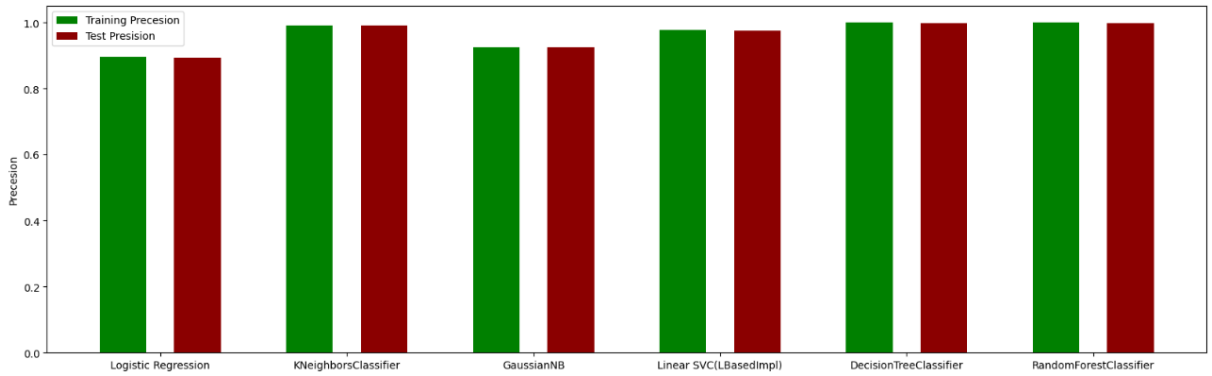

Figure 3.26: Training and testing precision of each model

Discussion: The figure 3.26 illustrates training and testing precision of each model, we can deduce from the figure that the training and test precision values reaching the maximum value of 1 for the decision tree classifier and random forest classifier indicate a high level of accuracy in correctly identifying positive instances. Since the precision remains high on the test data set as well, it suggests that the models generalize well and are not overfitting to the training data. Although the decision tree classifier and random forest classifier demonstrate the maximum precision value of 1, it is important to acknowledge the strong performance of the KNN, logistic regression, Naive bayes and SVM models. These models consistently achieve a commendable and higher recall value, showcasing their effectiveness in correctly capturing positive instances.

In the figure 3.27, we present the graph that displays both training and testing recall of each model:

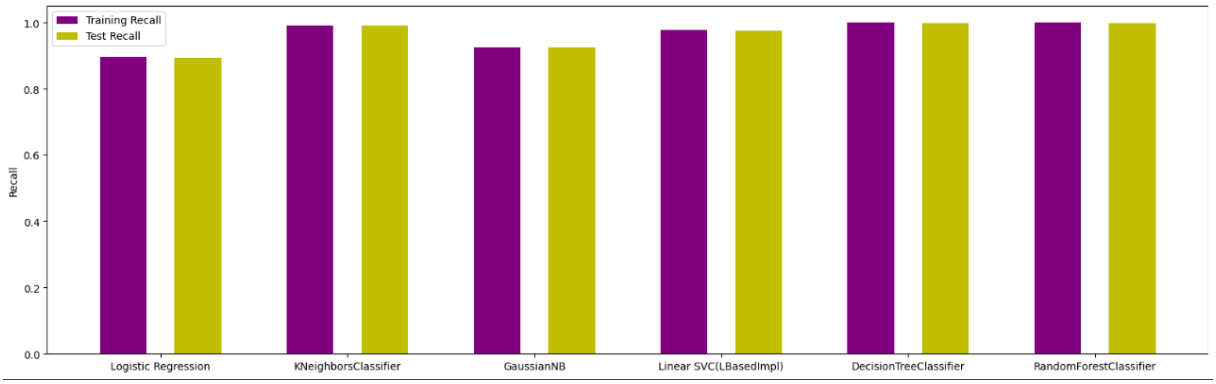

Figure 3.27: Training and testing recall of each model

Discussion: From the figure 3.27 that demonstrated the training and testing recall of each model, we can see that the decision tree classifier and the random forest classifier reaches the maximum recall value which is equal to 1, it indicates that these models have a low rate of false negatives, meaning they correctly identify almost all positive instances. The KNN, logistic regression, naive bayes and SVM models also exhibit notable performance in terms of recall, although they may not reach the maximum value of 1 as observed in the decision tree classifier and random forest classifier. These models consistently achieve a good and higher recall value, indicating their ability to correctly identify positive instances.

3.10 Conclusion:

In conclusion, this chapter has provided a comprehensive analysis of the performance of different models, including the logistic regression classifier, support vector machines classifier random forest classifier, KNN, SVM, and logistic regression, utilizing various evaluation metrics such as accuracy, precision, recall, ROC curves, and confusion matrices. the results and discussions in this chapter have provided valuable insights into the performance of various models, namely the decision tree classifier, random forest classifier, KNN, SVM, and logistic regression. The decision tree classifier and random forest classifier demonstrated exceptional performance across multiple evaluation metrics, including training and testing accuracy, precision, and recall. These models achieved maximum values, highlighting their ability to accurately classify instances. Furthermore, the logistic regression, Naive bayes, K-neighbors and support vector machines models also exhibited commendable performance. Although they did not reach the maximum values observed in the decision tree and random forest models, they consistently achieved higher values in both training and testing accuracy, precision, and recall. This signifies their effectiveness in accurately classifying instances and their capacity to generalize well to unseen data. In summary, the results and discussions presented in this chapter demonstrate the effectiveness of various models. These findings provide valuable insights for selecting the most appropriate model based on the specific requirements and characteristics of the data set.

General Conclusion

In this project, the objective was to develop an effective intrusion detection system utilizing various machine learning algorithms. The algorithms considered were Logistic Regression, Naive Bayes, K-Nearest Neighbors, Decision Tree, and Random Forest. The goal was to assess their performance in both training and testing phases and determine their effectiveness in accurately identifying and classifying network intrusions. Through the implementation and evaluation of these algorithms, it was observed that all of them exhibited good performance in terms of training and testing results. Each algorithm demonstrated the capability to learn from labeled data sets and effectively distinguish between normal and malicious network traffic patterns. The performance of the intrusion detection system was evaluated using standard evaluation metrics such as accuracy, precision, recall. The results indicated that all the considered algorithms achieved commendable performance, with high accuracy and balanced trade-offs between precision and recall. This project successfully demonstrated that utilizing machine learning algorithms, including Logistic Regression, Naive Bayes, K-Nearest Neighbors, Decision Tree, and Random Forest, can result in an effective intrusion detection system. The system exhibited good performance in both training and testing phases, showcasing the ability to accurately identify and classify network intrusions.

Bibliography

[1] Sevcan Yilmaz and Muhammet Nurullah Çeter. "Feature Selection and Comparison of Classification Algorithms for Intrusion Detection". In: *Journal of Network Security* (2018).

[2] Rebecca Gurley Bace, Peter Mell, et al. "Intrusion detection systems". In: (2001).

[3] Biswanath Mukherjee, L Todd Heberlein, and Karl N Levitt. "Network intrusion detection". In: *IEEE network* 8.3 (1994), pp. 26–41.

[4] Farzad Sabahi and Ali Movaghar. "Intrusion detection: A survey". In: *2008 Third International Conference on Systems and Networks Communications*. IEEE. 2008, pp. 23–26.

[5] ŞEREF SAĞIROĞLU, ESRA Yolacan, and URAZ YAVANOĞLU. "Designing and developing an intelligent intrusion detection system". In: *Journal of the Faculty of Engineering and Architecture of Gazi University* 26.2 (2011).

[6] Ming Liu et al. "Host-based intrusion detection system with system calls: Review and future trends". In: *ACM Computing Surveys (CSUR)* 51.5 (2018), pp. 1–36.

[7] Liquid Web. *Host-Based Intrusion Detection System*. Liquid Web. Oct. 2021.

[8] Giovanni Vigna and Richard A Kemmerer. "NetSTAT: A network-based intrusion detection approach". In: *Proceedings 14th Annual Computer Security Applications Conference (Cat. No. 98EX217)*. IEEE. 1998, pp. 25–34.

[9] Shreyansh Singh. "Network Intrusion Detection in an Adversarial Setting". In: (2019).

[10] Chibuzor John Ugochukwu, EO Bennett, and P Harcourt. *An intrusion detection system using machine learning algorithm*. LAP LAMBERT Academic Publishing, 2019.

[11] Hany Mohamed, Hesham Hefny, and Assem Alsawy. "Intrusion detection system using machine learning approaches". In: *Egypt. Comput. Sci. J* 42.3 (2018), pp. 1–13.

[12] Roman Fekolkin. "Intrusion Detection and Prevention Systems: Overview of Snort and Suricata". In: (January, 2015).

[13] Darren Mutz et al. "Anomalous system call detection". In: *ACM Transactions on Information and System Security (TISSEC)* 9.1 (2006), pp. 61–93.

[14] Bryan D Payne et al. "Lares: An architecture for secure active monitoring using virtualization". In: *2008 IEEE Symposium on Security and Privacy (sp 2008)*. IEEE. 2008, pp. 233–247.

[15] Gary C Kessler. "Defenses against distributed denial of service attacks". In: *SANS Institute* 2002 (2000).

[16] Chris Simmons et al. "AVOIDIT: A cyber attack taxonomy". In: *University of Memphis, Technical Report CS-09-003* (2009).

[17] José Tomás Martínez Garre, Manuel Gil Pérez, and Antonio Ruiz-Martínez. "A novel Machine Learning-based approach for the detection of SSH botnet infection". In: *Future Generation Computer Systems* 115 (2021), pp. 387–396.

[18] Felix C Freiling, Thorsten Holz, and Georg Wicherski. "Botnet tracking: Exploring a root-cause methodology to prevent distributed denial-of-service attacks". In: *Computer Security–ESORICS 2005: 10th European Symposium on Research in Computer Security, Milan, Italy, September 12-14, 2005. Proceedings 10*. Springer. 2005, pp. 319–335.

[19] Thomas M Chen and Cyrus Peikari. "Malicious software in mobile devices". In: *Handbook of Research on Wireless Security*. IGI Global, 2008, pp. 1–10.

[20] Ahmed Patel, Qais Qassim, and Christopher Wills. "A survey of intrusion detection and prevention systems". In: *Information Management & Computer Security* 18.4 (2010), pp. 277–290.

[21] Mark Dye, Rick McDonald, and Antoon Rufi. *Network Fundamentals, CCNA Exploration Companion Guide: CCNA Exploration Companion Guide*. Cisco press, 2007.

[22] Zoltán Dörnyei. *The psychology of the language learner: Individual differences in second language acquisition*. Routledge, 2014.

[23] Jafar Alzubi, Anand Nayyar, and Akshi Kumar. "Machine learning from theory to algorithms: an overview". In: *Journal of physics: conference series*. Vol. 1142. IOP Publishing. 2018, p. 012012.

[24] Shahadat Uddin et al. "Comparing different supervised machine learning algorithms for disease prediction". In: *BMC medical informatics and decision making* 19.1 (2019), pp. 1–16.

[25] Kevin M Mendez, Stacey N Reinke, and David I Broadhurst. "A comparative evaluation of the generalised predictive ability of eight machine learning algorithms across ten clinical metabolomics data sets for binary classification". In: *Metabolomics* 15 (2019), pp. 1–15.

[26] Iqbal H Sarker. "Machine learning: Algorithms, real-world applications and research directions". In: *SN computer science* 2.3 (2021), p. 160.

[27] Md Rafiul Hassan et al. "Early detection of cardiovascular autonomic neuropathy: A multi-class classification model based on feature selection and deep learning feature fusion". In: *Information Fusion* 77 (2022), pp. 70–80.

[28] André CPLF de Carvalho and Alex A Freitas. "A tutorial on multi-label classification techniques". In: *Foundations of Computational Intelligence Volume 5: Function Approximation and Classification* (2009), pp. 177–195.

[29] Ludwig Fahrmeir et al. *Regression models*. Springer, 2013.

[30] Hyeoun-Ae Park. "An introduction to logistic regression: from basic concepts to interpretation with particular attention to nursing domain". In: *Journal of Korean Academy of Nursing* 43.2 (2013), pp. 154–164.

[31] Mmm Dharmaraj. *Logistic Regression with StandardScaler-From the Scratch*. Medium Article. May 2022.

[32] Madan Somvanshi et al. "A review of machine learning techniques using decision tree and support vector machine". In: *2016 international conference on computing communication control and automation (ICCUBEA)*. IEEE. 2016, pp. 1–7.

[33] James Le. *The 10 Algorithms Machine Learning Engineers Need to Know*. Website. July 2016.

[34] Sourish Ghosh, Anasuya Dasgupta, and Aleena Swetapadma. "A study on support vector machine based linear and non-linear pattern classification". In: *2019 International Conference on Intelligent Sustainable Systems (ICISS)*. IEEE. 2019, pp. 24–28.

[35] skilltohire. *A Beginner's Guide to Parametric statistics*. Medium Article. July 2020.

[36] Jayalakshmi Rajamanickam et al. "Predictive model construction for prediction of soil fertility using decision tree machine learning algorithm". In: *INFOCOMP Journal of Computer Science* 20.1 (2021).

[37] MétéoSuisse-Blog. *L'IA au service de la prévision*. Blog de MétéoSuisse. Oct. 2022.

[38] Sadegh Bafandeh Imandoust, Mohammad Bolandraftar, et al. "Application of k-nearest neighbor (knn) approach for predicting economic events: Theoretical background". In: *International journal of engineering research and applications* 3.5 (2013), pp. 605–610.

[39] Kenza Harifi. *Bien comprendre l'algorithme des K-plus proches voisins (Fonctionnement et implémentation sur R et Python)*. Medium Article. Sept. 2019.

[40] K Ming Leung et al. "Naive bayesian classifier". In: *Polytechnic University Department of Computer Science/Finance and Risk Engineering* 2007 (2007), pp. 123–156.

[41] KDAS IIT KGP. *Naive Bayes Algorithm*. Medium Article. Sept. 2021.

[42] John M McNamara, Richard F Green, and Ola Olsson. "Bayes' theorem and its applications in animal behaviour". In: *Oikos* 112.2 (2006), pp. 243–251.

[43] Zoubin Ghahramani. "Unsupervised learning". In: *Summer school on machine learning*. Springer, 2003, pp. 72–112.

[44] Amandeep Kaur Mann and Navneet Kaur. "Review paper on clustering techniques". In: *Global Journal of Computer Science and Technology* (2013).

[45] Anurag Chouhan. "K-means Clustering & its Real use-case in the Security Domain". In: *LinkedIn* (July 2021).

[46] Sotiris Kotsiantis and Dimitris Kanellopoulos. "Association rules mining: A recent overview". In: *GESTS International Transactions on Computer Science and Engineering* 32.1 (2006), pp. 71–82.

[47] Mohammad Khubeb Siddiqui and Shams Naahid. "Analysis of KDD CUP 99 dataset using clustering based data mining". In: *International Journal of Database Theory and Application* 6.5 (2013), pp. 23–34.

[48] Sathyanarayanan Revathi and A Malathi. "A detailed analysis on NSL-KDD dataset using various machine learning techniques for intrusion detection". In: *International Journal of Engineering Research & Technology (IJERT)* 2.12 (2013), pp. 1848–1853.

[49] Huilong Ao. "Using machine learning models to detect different intrusion on NSL-KDD". In: *2021 IEEE International Conference on Computer Science, Artificial Intelligence and Electronic Engineering (CSAIEE)*. IEEE. 2021, pp. 166–177.

[50] Onur Rauf Bingol and Adarsh Krishnamurthy. "NURBS-Python: An open-source object-oriented NURBS modeling framework in Python". In: *SoftwareX* 9 (2019), pp. 85–94.

[51] David Wiley. "Open source, openness, and higher education". In: *Innovate: Journal of Online Education* 3.1 (2006).

[52] Toon Calders, Bart Goethals, and Szymon Jaroszewicz. "Mining rank-correlated sets of numerical attributes". In: *Proceedings of the 12th ACM SIGKDD international conference on knowledge discovery and data mining*. 2006, pp. 96–105.

[53] Rajiv D Banker and Richard C Morey. "The use of categorical variables in data envelopment analysis". In: *Management science* 32.12 (1986), pp. 1613–1627.

[54] Karen A Monsen and Karen A Monsen. "Descriptive Analysis and Interpretation". In: *Intervention Effectiveness Research: Quality Improvement and Program Evaluation* (2018), pp. 53–62.

Appendices

.1 Libraries importation

```
import numpy as np
import pandas as pd
import warnings
import matplotlib.pyplot as plt
import seaborn as sns
import tensorflow as tf
from tensorflow.keras import regularizers
from sklearn.naive_bayes import GaussianNB
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.preprocessing import RobustScaler
from sklearn.ensemble import RandomForestClassifier, RandomForestRegressor
from sklearn.model_selection import train_test_split
from sklearn import svm
from sklearn import metrics
pd.set_option('display.max_columns',None)
warnings.filterwarnings('ignore')
%matplotlib inline
```

.2 Data importation

To import the NSL-KDD dataset into Google Colab, we use the wget command that is used to download the dataset's ZIP file from the provided URL:

```
[2]  !wget "http://205.174.165.80/CICDataset/NSL-KDD/Dataset/NSL-KDD.zip"
```

.3 Descriptive analysis

To gain insights into the numerical features of the dat aset, a statistical analysis was conducted using the describe() method from the pandas library. This method calculated various descriptive statistics, including count, mean, standard deviation, minimum, quartiles, and maximum values for each numerical column in the data_train DataFrame as follows:

```
[11] data_train.describe().style.background_gradient(cmap='Greens').set_properties(**{'font-family':'Segoe UI'})
```

.4 Testing data balance

To conduct a data balance test, I applied two lines of code as shwon in the figure below:

```
[50] data_train.loc[data_train['outcome'] == "normal", "outcome"] = 'normal'
     data_train.loc[data_train['outcome'] != 'normal', "outcome"] = 'attack'
```

By executing these two lines of code, we transformed the original multi-class classification problem into a binary classification problem. This allowed us to focus on distinguishing between 'normal' and 'attack' instances rather than considering specific attack types.